**Selenium:**

This is used to automate web browser.

**4 major components:**

1. Selenium IDE 🡪 Record and playback
2. Selenium RC 🡪 Deprecated from the market
3. Selenium Webdriver 🡪 latest version 3.x
4. Selenium Grid 🡪

**Limitations:**

Supports only web application and since it is an open source. So, you will not get any support when you face any issue, but we have so many blogs, forums / communities to discuss

**Advantage:**

Platform independent and works on multiple browser

Open source

**Jar Files:**

1. **J**ava **ar**chive files which are like a zipped file, and which contains one or more class that can be reused in another project
2. Basically, it contains a .class files (not a readable format and it is in byte code, meta data, and some other resources

**How to create a new Jar file:**

1. Create a project and right click on it and say export
2. Select Java 🡪 Jar 🡪 next🡪 Select a destination to save the file 🡪 Next 🡪 Next 🡪 select the main class and click Finish

**To reuse the jar file in another class:**

1. Right click on a project 🡪 Build Path 🡪 Configure Build Path 🡪 Libraries 🡪 Class path 🡪 Add external jars
2. After the jars are, instantiate the class (the class used inside the jar) in the new class and call the method.

**How to run the jar using command prompt:**

1. Open the command prompt from the jar location 🡪 java jar [jarname] [packagename.classname]

**When to use jar and runnable jar file:**

1. When you want to run a program without IDE, jar file is the best choice. After creating a jar, we use the command prompt to run the jar as mentioned above
2. If the jar that we are going to run has an interdependency on another jar, then we need to use create runnable jar for that project instead of normal jar.

**Installation:**

<https://www.selenium.dev/downloads/>

**Selenium Architecture:**

Diagram

Description automatically generated

**WebDriver Interface:**

WebDriver is a remote-control interface that enables introspection and control of user agents (browsers).

* SearchContext is the super most interface in selenium, which is extended by another interface called WebDriver.
* All the abstract methods of SearchContext and WebDriver interfaces are implemented in RemoteWebDriver class.
* All the browser related classes such as FirefoxDriver, ChromeDriver etc., extends the RemoteWebdriver class.

WebDriver defines common methods which all browser classes (such as Firefox, Chrome etc.,) use. All these class methods are derived from WebDriver interface.

All the [abstract](https://www.softwaretestingmaterial.com/abstraction-in-java/) methods of both the [interfaces](https://www.softwaretestingmaterial.com/interface-in-java/) are implemented in RemoteWebDriver class which is extended by browser classes such as Firefox Driver, Chrome Driver etc.

**What is a web element?**

Web elements are nothing but HTML elements like textbox, dropdowns radio buttons, submit buttons, etc. These HTML elements are written with **start** tag and ends with an **end** tag.

**XPath in Selenium:**

XML Path used for navigation through the HTML structure of the page. It is a syntax or language for finding any element on a web page using XML path expression.

Diagram

Description automatically generated

**2 types of XPath**

1. Absolute XPath
2. Relative XPath

Demo site: <http://demo.guru99.com/test/selenium-xpath.html>



Find the XPath for the field “Testing” in the above link as shown below

**Absolute Path** 🡪 “/html/body/div[4]/div[1]/div/h4[1]”

**Relative Path** 🡪 //b[contains(text(),'Testing')]

**XPath axes:**

XPath axes search different nodes in XML document from current context node. XPath Axes are the methods used to find dynamic elements, which otherwise not possible by normal XPath method having no ID, Classname, Name, etc.

1. **Basic XPath:**

For the field “UserID” in the above link

XPath=//input[@name='uid']

XPath=//input[@type='text']

1. **Contains:**

Contains() is a method used in XPath expression. It is used when the value of any attribute changes dynamically, for example, login button

The **contain** feature has an ability to find the element with partial text

XPath 🡪 //\*[contains(@type,'sub')] or //input[contains(@type,’sub’] or //\*[contains(@value,'LOG')]

With Full text

Xpath=//\*[contains(@name,'btn')] or //\*[contains(@name,'btnLogin')]

1. **Using OR & AND:**

For this example let us take the “Login” and “Reset” button

XPath=//\*[@type='submit' or @name='btnReset'] 🡪 both login and reset button will get highlighted

XPath = //input[@type='reset' and @name='btnReset'] 🡪 only reset button will get highlighted

1. **XPath using Starts-with:**

**XPath starts-with()** is a function used for finding the web element whose attribute value gets changed on refresh or by other dynamic operations on the webpage. For example, if you have the locator id has value as shown below, then XPath will be

Id=” message12″

Id=” message345″

XPath=//label[starts-with(@id,'message')]

1. **XPath using Text():**

The field label “UserID” can be found using the following xpath

Xpath=//td[text()='UserID']

1. **XPath axes methods:**

These XPath axes methods are used to find the complex or dynamic elements.

1. **Following:**

Selects all elements in the document of the current node( ) [ UserID input box is the current node] as shown in the below screen.

Xpath=//\*[@type='text']//following::input

Note: When we get multiple nodes for the same, then use the node number as in

Xpath=//\*[@type='text']//following::input[1]

1. **Ancestor:**

The ancestor axis selects all ancestor’s element (grandparent, parent, etc.) of the current node as shown in the below screen.

In the below expression, we are finding ancestors element of the current node(“ENTERPRISE TESTING” node)

Xpath=//\*[text()='Enterprise Testing']//ancestor::div

1. **Child:**

Selects all children elements of the current node (Java) as shown in the below screen.

Xpath=//\*[@id='java\_technologies']//child::li

1. **Preceding:**

Select all nodes that come before the current node as shown in the below screen

In the below expression, it identifies all the input elements before “LOGIN” button that is **Userid** and **password** input element.

Xpath=//\*[@type='submit']//preceding::input

1. **Following-Sibling:**

Select the following siblings of the context node. Siblings are at the same level of the current node as shown in the below screen. It will find the element after the current node.

xpath=//\*[@type='submit']//following-sibling::input

1. **Parent:**

Selects the parent of the current node as shown in the below screen.

Xpath=//\*[@id='rt-feature']//parent::div

Rest of the stuff 🡪 please go through <https://www.guru99.com/xpath-selenium.html#basic-xpath>

**WebElement Commands:**

Command used to perform on web element.

For demo use 🡪 <https://www.browserstack.com/guide/selenium-webelement-commands>

**Types of Web element command:**

1. [sendKeys() command](https://www.browserstack.com/guide/sendkeys-in-selenium) allows the user to type content automatically into an editable field
2. [isDisplayed() command](https://www.browserstack.com/guide/isdisplayed-method-in-selenium) in Selenium verifies if a certain element is present and displayed.
3. isSelected() command only works on input elements such as radio buttons, checkboxes, select options, and menu items. It is used to determine if an element is selected.
4. Submit() command is used to submit a webpage like pressing enter key in keyboard
5. isEnabled() command is used to check a particular web element is enabled to perform operation on it
6. clear() command is used to clear the text entered in the editable field
7. getText() command is used to get the text within a specific web element.
8. getTagName() command is used to get the tag name of a specific web element
9. getCssValue() command is used to retrieves the CSS property value of a specified element.
10. getAttribute() command is used to get the attribute of the particular web element
11. click() command is used to perform a click operation on a web element

**Dynamic Table:**

The row and columns are not fixed and keeps changes every time when we load the web page.

reference 🡪 <https://www.guru99.com/handling-dynamic-selenium-webdriver.html>

For demo 🡪 <http://demo.guru99.com/test/web-table-element.php>

**How to get the number of rows and columns from dynamic web table:**

1. instantiate the driver and open the web page
2. get the number of rows using findelements by with xpath = ‘//table//tr’ with return type as List<WebElement>
3. Print the number of rows using size method on list
4. Similarly find the columns using xpath=’//table//th’

**How to fetch a cell value of a particular row and column of the dynamic table:**

1. Instantiate the driver and open the web page
2. Get the 3rd row using xpath = “//table//tr[3]”
3. Get the 3rd element in 3rd using xpath = “//tbody/tr[3]/td[4]”

**Drop down with multi select option:**

For Demo 🡪 <https://www.tutorialspoint.com/selenium/selenium_automation_practice.htm>

1. Instantiate the driver and open the web page
2. Identify the dropdown using xpath = ‘"//select[@name='selenium\_commands']"’ and assign it to a web element named dropDownlist
3. To get the options of dropdown use the drop down web element and dropDownlist.getOptions() and assign it to a list.
4. To check whether multi select is enabled use dropDownlist.isMultiple()
5. To select by index, say dropDownlist.selectByIndex(2)
6. To select by value, say dropDownlist.selectByValue(“option 1”)
7. To deselect by visibleText say dropDownlist.deselectByVisibleText("Selenium")

**Waits in Selenium:**

Waits are generally used to make our code robust and reliable.

1. **Implicitly wait**

**Selenium 3.x**

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

**Selenium 4**

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

1. **Pageload timeout**

driver.manage().timeouts().scriptTimeout(Duration.ofMinutes(2)); driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(10));

1. **Fluent wait**

**//Selenium 3**

Wait<WebDriver> wait = new FluentWait<WebDriver>(driver).withTimeout(30, TimeUnit.SECONDS)

.pollingEvery(5, TimeUnit.SECONDS)

.ignoring(NoSuchElementException.class);

**//Selenium 4**

Wait<WebDriver> fluentWait = new fluentWait<WebDriver>(driver) .withTimeout(Duration.ofSeconds(30))

.pollingEvery(Duration.ofSeconds(5))

.ignoring(NoSuchElementException.class);

1. **Explicit wait**

//Selenium 3

WebDriverWait wait = new WebDriverWait(driver,10); wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector(".classlocator")));

//Selenium 4

WebDriverWait wait = new WebDriverWait(driver,Duration.ofSeconds(10)); wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector(".classlocator")));

**Alerts in Selenium:**

For Ref 🡪 <https://www.guru99.com/alert-popup-handling-selenium.html>

For demo 🡪 <https://www.w3schools.com/jsref/met_win_prompt.asp>

Achieve the following using the above demo link

1. Launch the url
2. Click on a button to open a new window
3. Using switch to window jump to the newly opened tab
4. Using switch to frame jump to the desired frame to click a button
5. On the prompt, using send key, send a text and accept it
6. Back to the parent frame
7. Back to the parent window and perform a click operation

**Keyboard Events:**

It can either be achieved using sendKeys(), Robot class (using keypress(VK\_keyname) or keyrelease(VK\_keyname) and actions class.

Note: key release is used when we deal with shift, ctrl, alt etc.,

**Using Robot class**

Demo 🡪 <https://www.toolsqa.com/selenium-webdriver/robot-class-keyboard-events/>

**Using Actions Class**

Sendkeys has limitation when it comes to perform some keyboard actions like copy and paste using (Ctrl+c/V). Though we write the code as sendkeys(keys.ctrl), sendkeys(C), it fails. Hence we should use actions class

Actions myAction, Action seriesofActions;

myAction = new Actions(driver);

seriesofAction = myAction.moveToElement(driver.FindElement(By.id(“”)).click().sendKeys(“”). build();

seriesofAction.perform();

**How to perform double click:**

Actions action = new Actions(driver);

WebElement element = driver.findElement(By.id(“”));

Action.doubleClick(element).perform();

**How to perform right click:**

Actions action = new Actions(driver);

WebElement element = driver.findElement(By.id(“”));

Action.contextClick(element).perform();

**How to perform drag and drop:**

Actions action = new Actions(driver);

WebElement element = driver.findElement(By.id(“”));

WebElement element1 = driver.findElement(By.id(“”));

Action.dragAndDrop(element,element1).build().perform();

**Select the Current Address using CTRL + A**

Actions action = new Actions(driver);

action.keyDown(Keys.CONTROL); instead of control we can use Alt or Shift too

action.sendKeys("a");

action.keyUp(Keys.CONTROL);

action.build().perform();

Similarly for copy change the second line to action.sendKeys("c") and for paste change it to action.sendKeys("v");

**For tab out:**

actions.sendKeys(Keys.TAB);

actions.build().perform();

**Browser Profiles:**

**How to set firefox profiles:**

ProfilesIni profile = new ProfilesIni(); //obj for firefox profile

FirefoxProfile myProfile = profile.getProfile(“profile”);

WebDriver driver = new FirefoxDriver(myprofile);

**Other way of doing it:**

// Create object for Firefox Profile

FirefoxProfile myprofile=new FirefoxProfile(newFile("\c:users\AppData\MozillaFirefoxProfile\_name.default "));

//Initialize Firefox driver

WebDriver driver = new FirefoxDriver(myprofile);

**How to achieve this in chrome browser:**

System.setProperty("webdriver.chrome.driver","C:\\chromedriver.exe");

ChromeOptions options = new ChromeOptions();

options.addArguments("user-data-dir=C:/Users/user\_name/AppData/Local/Google/Chrome/User Data");

options.addArguments("--start-maximized");

driver = new ChromeDriver(options);

**How to perform automation on a manually opened browser:**

First open the command prompt and type the following command and press Enter

**Note**: 1. No spaces between 2 hypens and chrome executable should have been added to run environment

chrome.exe - -remote-debugging-port=9222 - - user-data-dir=”C:\Selenium\Automaionprofile” and press enter

The above command will open the google chrome and to check whether the opened chrome is from the command prompt then open a new chrome and enter the url as “IPaddressOfTheSys:9222” and press enter. Click inspectable page. Both browsers will show the same screen.

In Eclipse:

ChromeOptions option = new ChromeOptions();

Option.setExperimentalOption(“debuggerAddress”,”ipaddress:9222”);

WebDriver driver = new Chromedriver(option);

**How to handle SSL:**

For ref 🡪 <https://www.guru99.com/ssl-certificate-error-handling-selenium.html>

**For TestNG Reporter:**

After testng class is run, open the test output folder and double click “index.html” or “emailable-report.html”