1) Selenium Form WebElement: TextBox, Submit Button, sendkeys(), click()

* Text

This method will fetch the visible (i.e. not hidden by CSS) innerText of the element. This accepts nothing as a parameter but returns a String value.

Eg : String linkText = element.getText();

* Submit

This method works well/better than the click() if the current element is a form, or an element within a form. This accepts nothing as a parameter and returns nothing.

Eg : driver.findElement(By.id("SubmitButton")).submit();

* sendkeys()

This simulates typing into an element, which may set its value. This method accepts CharSequence as a parameter and returns nothing.

Eg : driver.findElement(By.id("UserName")).sendKeys("ToolsQA");

* click()

This simulates the clicking of any element. Accepts nothing as a parameter and returns nothing.

Eg : driver.findElement(By.linkText("ToolsQA")).click();

2) How to Select CheckBox and Radio Button in Selenium WebDriver

* Radio Button

Radio Buttons too can be toggled on by using the click() method.

Eg : WebElement radio = driver.findElementById(“QA”);

radio.click();

* Check Box

Toggling a check box on/off is also done using the click() method.

Eg : WebElement check = driver.findElementById(“QA”);

for(int i=0;i<2;i++) {

check.click();

}

3) How to Click on Image in Selenium Webdriver

Image links are the links in web pages represented by an image which when clicked navigates to a different window or page.Since they are images, we cannot use the By.linkText() and By.partialLinkText() methods because image links basically have no link texts at all.

In this case, we should resort to using either By.cssSelector or By.xpath. The first method is more preferred because of its simplicity.

Eg : driver.findElement(By.cssSelector("a[title=\"Go to Facebook home\"]")).click();

4) How to Select Value from DropDown using Selenium Webdriver (Repeated Question)

5) Mouse Click & Keyboard Event: Action Class in Selenium Webdriver

|  |  |
| --- | --- |
| **Method** | **Description** |
| **clickAndHold()** | Clicks (without releasing) at the current mouse location. |
| **contextClick()** | Performs a context-click at the current mouse location. (Right Click Mouse Action) |
| **doubleClick()** | Performs a double-click at the current mouse location. |
| **dragAndDrop(source, target)** | Performs click-and-hold at the location of the source element, moves to the location of the target element, then releases the mouse.  **Parameters:**  source- element to emulate button down at.  target- element to move to and release the mouse at. |
| **dragAndDropBy(source, x-offset, y-offset)** | Performs click-and-hold at the location of the source element, moves by a given offset, then releases the mouse.  **Parameters**:  source- element to emulate button down at.  xOffset- horizontal move offset.  yOffset- vertical move offset. |
| **keyDown(modifier\_key)** | Performs a modifier key press. Does not release the modifier key - subsequent interactions may assume it's kept pressed.  **Parameters**:  modifier\_key - any of the modifier keys (Keys.ALT, Keys.SHIFT, or Keys.CONTROL) |
| **keyUp(modifier \_key)** | Performs a key release.  **Parameters**:  modifier\_key - any of the modifier keys (Keys.ALT, Keys.SHIFT, or Keys.CONTROL) |
| m**oveByOffset(x-offset, y-offset)** | Moves the mouse from its current position (or 0,0) by the given offset.  **Parameters**:  x-offset- horizontal offset. A negative value means moving the mouse left.  y-offset- vertical offset. A negative value means moving the mouse down. |
| **moveToElement(toElement)** | Moves the mouse to the middle of the element.  **Parameters**:  toElement- element to move to. |
| **release()** | Releases the depressed left mouse button at the current mouse location |
| **sendKeys(onElement, charsequence)** | Sends a series of keystrokes onto the element.  **Parameters**:  onElement - element that will receive the keystrokes, usually a text field  charsequence - any string value representing the sequence of keystrokes to be sent |

6) How to Upload & Download a File using Selenium Webdriver

* File Upload

**Uploading files in WebDriver is done by simply using the sendKeys() method on the file-select input field to enter the path to the file to be uploaded.**

Remember following two things when uploading files in WebDriver

* There is no need to simulate the clicking of the "Browse" button. WebDriver automatically enters the file path onto the file-selection text box of the <input type="file"> element
* When setting the file path in your Java IDE, use the proper escape character for the back-slash.
* File Download

**WebDriver has no capability to access the Download dialog boxes**presented by browsers when you click on a download link or button. However, we can bypass these dialog boxes using a separate program called "wget".

* **Wget is a small and easy-to-use command-line program used to automate downloads**. Basically, we will access Wget from our WebDriver script to perform the download process.

7) Robot Class in Selenium Webdriver

In certain Selenium Automation Tests, there is a need to control keyboard or mouse to interact with OS windows like Download pop-up, Alerts, Print Pop-ups, etc. or native Operation System applications like Notepad, Skype, Calculator, etc. Selenium Webdriver cannot handle these OS pop-ups/applications.

* Benefits of Robot class
* Robot Class can simulate Keyboard and Mouse Event
* Robot Class can help in upload/download of files when using selenium web driver
* Robot Class can easily be integrated with current automation framework (keyword, data-driven or hybrid)

8) Handling iFrames in Selenium Webdriver: switchTo() (Repeated Question)

9) Implicit, Explicit, & Fluent Wait in Selenium WebDriver

* Implicit

Implicit Wait will let Selenium WebDriver to wait for a certain amount of time, before throwing the **exception** that it cannot find the element on the web page.

Note: The Implicit Wait will be in place, when the browser is opened. This means, that any search for elements on the page could take the time the Implicit Wait is set for.

Eg : WebDriver driver = new FirefoxDriver();

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

driver.get("myURL");

WebElement myDynamicElement = driver.findElement(By.id("myDynamicElement"));

* Explicit

It is more extendable in the means that you can set it up to wait for any condition you might like. Usually, you can use some of the pre-built Expected Conditions to wait for elements to become clickable, visible, invisible, etc.

Eg : driver.manage().timeouts().implicitlyWait(0, TimeUnit.SECONDS); //nullify implicitlyWait()

WebDriverWait wait = new WebDriverWait(driver, timeOutInSeconds);

element = wait.until(ExpectedConditions.visibilityOfElementLocated(by));

* Fluent

Each FluentWait instance defines the maximum amount of time to wait for a condition, as well as the frequency with which to check the condition. Furthermore, the user may configure the wait to ignore specific types of exceptions while waiting, such as NoSuchElementExceptions when searching for an element on the page.

Eg : FluentWait wait = new FluentWait(driver).withTimeout(timeOutInSeconds,TimeUnit.SECONDS)

.pollingEvery(200,TimeUnit.MILLISECONDS)

.ignoring(NoSuchElementException.class);

element = (WebElement) wait.until(ExpectedConditions.visibilityOfElementLocated(by));

10) Double click and Right Click in Selenium with Examples

* Double Click

Double click action in Selenium web driver can be done using Actions class. Actions class is a predefined class in Selenium web driver used to perform multiple keyboard and mouse operations such as Right Click, Drag and Drop, etc.

Eg : Double click in Selenium using Actions class

Actions actions = new Actions(driver);

WebElement elementLocator = driver.findElement(By.id("ID"));

actions.doubleClick(elementLocator).perform();

* Right click

Right click action in Selenium web driver can be done using Actions class. Right Click operation is also called Context Click in Selenium. Pre-defined method context click provided by Actions class is used to perform right click operation.

Eg : Double click in Selenium using Actions class

Actions actions = new Actions(driver);

WebElement elementLocator = driver.findElement(By.id("ID"));

actions.contextClick(elementLocator).perform();