Locators:

- Id

- name

- className

- tagName

- linkText

- partialLinkText

- cssSelector

- xpath

How to use locators:

driver.findElement(By.id(“abc”));

driver.findElement(By.xpath(“//input[@class=’abc’]”));

How to decide which locator to use:

1. If element is a static and unique link - choose linkText. Example "Sign In"

2. If element is a dynamic link with some static part - choose partialLinkText.

Example "Welcome user 133"

3. If element has id and it is static and unique - choose id

4. If element has name and it is static and unique - choose name

5. If element has class and it is static and unique - choose class

Beware that UI developer's use class to give look and feel to element (width, font, color etc). So, multiple elements may have same class.

6. If you want to use combination of tag and attributes like class or id to locate element - choose cssSelector

7. If you want to use combination of tag and multiple attributes to locate element - choose xpath

8. If tag is static and unique for element you want to locate - Choose tagName

Launching browser:

Firefox :

```
WebDriverManager.FirefoxDriver().setup();

WebDriver driver = new FirefoxDriver();
```

Chrome :

```
WebDriverManager.ChromeDriver().setup();

WebDriver driver = new ChromeDriver();
```

IE :

```
WebDriverManager.InternetExplorerDriver().setup();

WebDriver driver = new InternetExplorerDriver();
```

Navigating to url:

Method 1: *driver.get("http://google.com");*

Method 2: *driver.navigate().to("http://google.com");*

Implicit Wait:

Specifies the amount of time the driver should wait when searching for an element if it is not immediately present.

Example:

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

WebElement:

Every element on HTML page (text filed, radio button, check box, button, label, list box etc.) is represented by object of WebElement.

driver.findElement method:

- Finds the first WebElement with given locator and returns WebElement object for first matching element.

- Throws NoSuchElementException if no matching elements are found

Example:

```
WebElement ele = driver.findElement(By.id("host"));
```

driver.findElements method:

- Finds all WebElements with given locator and returns list of WebElement objects.

- Return empty list if no matching elements are found

Example:

List<WebElement> allEles = driver.findElements(By.id("host"));

allEles.get(0).sendKeys(“Test”); // type Test into first WebElement of list

UI Operations:

Click:

WebElement ele = driver.findElement(By.id("host")); ele.click();

Type:

WebElement ele = driver.findElement(By.id("host")); ele.clear(); // clears text field

ele.sendKeys("Test"); // enters value in text field

Read text:

WebElement ele = driver.findElement(By.id("host"));

String value = ele.getText();

Is element displayed:

WebElement ele = driver.findElement(By.id("host"));

boolean isDisplayed = ele.isDisplayed();

Is element enabled:

WebElement ele = driver.findElement(By.id("host"));

boolean isEnabled = ele.isEnabled();

Is element selected (applicable to checkbox, radiobutton and listbox)**:**

WebElement ele = driver.findElement(By.id("host"));

boolean isSelected = ele.isSelected();

Select element(s) in list box:

WebElement ele = driver.findElement(By.id("host"));

Select list = new Select(ele);

list.selectByIndex(3); // selects element with index attribute 3
list.selectByVisibleText("Mr"); // selects element by text displayed in UI
list.selectByValue("test"); // selects element with value attribute 'test'

Xpath:

Language used to locate elements in xml/html document

Types:

Absolute: Starts from root element (html)

Relative: Starts with //tag-name

Using attributes:

Syntax: //tag-name [@ attribute = 'value'][@ attribute2 = 'value2']

E.g. *//input[@name='name'][@title='Name for this connection']*

Using partial attribute:

Syntax: //tag-name [contains(@ attribute, 'partial value')]

E.g. *//input[contains(@title, 'Name for')]*

To choose specific element out of multiple

Syntax: (// xpath) [index]

E.g. (//input[@type='text'])[2]

To go to parent tag

Syntax: ..

E.g. //input[contains(@title, 'Name for')]/..

To go to direct child

Syntax: /

E.g. (//form[@name='connectForm']/fieldset/div)[2]/div/input

locate child at any level

Syntax: //

E.g. //form[@name='connectForm']//input

Following Siblings

Syntax: //xpath/following-sibling::tag-name

E.g. //label[contains(@title, 'Name for')]/following-sibling::div/input

Preceding Siblings

Syntax: //xpath/preceding-sibling::tag-name

E.g. //input[@name='name']/../preceding-sibling::label

To locate tag using text

Syntax: use . or text() instead of @attribute

E.g. //tag-name[. = 'text-outside-brackets']

//label[.='Host: ']

//label[contains(., 'Host:')]

//tag-name[text() = 'text-outside-brackets

Browser Operations:

Maximize browser window:

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

To refresh page:

driver.navigate().refresh();

To click back button of browser:

driver.navigate().back();

To click forward button of browser:

driver.navigate().forward();

Quit and Close: driver.quit();

Close all browser windows opened by your Selenium program.

It also marks driver object as dead. You cannot call any driver function after quit.

It will fail at runtime.

driver.close();

Close only active browser windows opened by your Selenium.

But you can still call driver functions after close.

Switching between windows:

Selenium identifies browsers using a string called window handle.

To get handle of currently active window:

String handle = driver.getWindowHandle();

To get handles of all browser windows opened by Selenium:

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

To switch to window:

driver.switchTo().window(name or handle);

Handling javascript alerts:

Alert alert = driver.switchTo().alert(); alert.accept(); // to click OK

alert.dismiss(); // to click Cancel alert.sendKeys(); // to type a value