Comprehensive Guide to Selenium WebDriver and SearchContext Methods

1 get()

- **Declared in:** WebDriver interface
- **Implemented by:** RemoteWebDriver class
- To call this non-static method, create an object of WebDriver interface.
- **Arguments:** Takes a URL as a String.
- **Returns:** void (does not return anything).
- **Purpose:** Launches a website.

```
WebDriver driver = new ChromeDriver();
driver.get("https://www.example.com");
```

2. getTitle()

- **Declared in:** WebDriver interface
- **Implemented by:** RemoteWebDriver class
- To call this non-static method, create an object of WebDriver interface.
- **Arguments:** None.
- **Returns:** Title of the page as String type.
- **Purpose:** Useful to get the title of the current page.

```
WebDriver driver = new ChromeDriver();
 String title = driver.getTitle();
```

3.getPageSource()

- **Declared in**: WebDriver interface
- **Implemented by:** RemoteWebDriver class
- To call this non-static method, create an object of WebDriver interface.
- **Argument**: None.
- **Returns**: Source code of the page as String type.
- **Purpose**: Useful to get the source code of the current page

```
WebDriver driver = new ChromeDriver();
String pageSource = driver.getPageSource();
```

4. getCurrentUrl()

- **Declared in**: WebDriver interface
- **Implemented by:** RemoteWebDriver class
- To call this non-static method, create an object of WebDriver interface.
- **Argument**: None.
- **Returns**: URL of the page as String type.
- **Purpose**: Useful to get the URL of the current page.

```
WebDriver driver = new ChromeDriver();
String currentUrl = driver.getCurrentUrl();
```

5. getWindowHandle()

- **Declared in**: WebDriver interface
- **Implemented by:** RemoteWebDriver class
- To call this non-static method, create an object of WebDriver interface.
- **Argument**: None.
- **Returns**: Handle value of the current browser window/tab as String type.
- **Purpose**: Useful to get the handle value of the current browser window/tab.

```
WebDriver driver = new ChromeDriver();
String windowHandle = driver.getWindowHandle();
```

6. getWindowHandles()

- **Declared in**: WebDriver interface
- **Implemented by:** RemoteWebDriver class
- To call this non-static method, create an object of WebDriver interface.
- **Argument**: None.
- **Returns**: Handle values of all open browser windows/tabs as Set<String>.
- **Purpose**: Useful to get handle values of all open browser windows/tabs.

```
WebDriver driver = new ChromeDriver();
Set<String> Handles = driver.getWindowHandle();
```

7. close()

- **Declared in**: WebDriver interface
- **Implemented by:** RemoteWebDriver class
- To call this non-static method, create an object of WebDriver interface.
- **Argument**: None.
- **Returns**: void.
- **Purpose**: Useful to close the current browser window/tab.

```
WebDriver driver = new ChromeDriver();
driver.close();
```

8. quit()

- **Declared in**: WebDriver interface
- **Implemented by:** RemoteWebDriver class
- To call this non-static method, create an object of WebDriver interface.
- **Argument**: None.
- **Returns**: void.
- **Purpose**: Useful to close all browser windows/tabs.

```
WebDriver driver = new ChromeDriver();
 driver.quit();
```

9. navigate()

- **Declared in**: WebDriver interface
- **Implemented by**: RemoteWebDriver class

- To call this non-static method, create an object of WebDriver interface.
- **Argument**: None.
- **Returns**: Object of Navigation interface.
- **Purpose**: Useful to navigate through the browser (back, forward, refresh, navigate to URL).

```
WebDriver driver = new ChromeDriver();
driver.navigate().to("https://www.google.com");
driver.navigate().back();
driver.navigate().forward();
driver.navigate().refresh();
```

10. switchTo()

- **Declared in**: WebDriver interface
- **Implemented by**: RemoteWebDriver class
- To call this non-static method, create an object of WebDriver interface.
- **Argument**: None.
- **Returns**: Object of TargetLocator interface.
- **Purpose**: Useful to move the focus of the driver to a specific frame, alert, or active element on the current page.

i. SwitchTo().frame()

Frames are HTML documents embedded inside another document. Use this method to interact with elements inside them.

- **Purpose**: Shifts the focus of the driver to a specific frame on the webpage.
- **Sub-Methods**:
    - driver.switchTo().frame(String frameNameOrId) – Switch using the frame's name or ID.
    - driver.switchTo().frame(int frameIndex) – Switch using the index (0-based).
    - driver.switchTo().frame(WebElement frameElement) – Switch using a WebElement representing the frame.
    - driver.switchTo().parentFrame() – Switch back to the parent frame.
    - driver.switchTo().defaultContent() – Return to the top-level page content.

ii. switchTo().alert()

Alerts are browser pop-ups requiring user interaction.

- **Purpose**: Switches the focus to an alert box on the page.
- **Actions on Alerts**:
    - **.accept():** Clicks the "OK" button on the alert.
    - **.dismiss():** Clicks the "Cancel" button on the alert.
    - **.getText():** Retrieves the alert's text.
    - **.sendKeys(String keysToSend)**: Sends input to the alert box (for prompts).

```
Alert alert = driver.switchTo().alert();
System.out.println(alert.getText());
alert.accept(); // Dismissing an alert
alert.dismiss(); // Dismissing the alert
alert.sendKeys("Test Input"); // Sending input to a prompt alert
driver.switchTo().alert().accept(); // Accepting the alert
```

iii. switchTo().window()

Handles switching between multiple browser tabs or windows.

- **Purpose**: Switches the focus to a specific browser window or tab using its handle.

Sub-Methods:

- driver.switchTo().window(String windowHandle) – Switch to a window using its handle.
- driver.getWindowHandle() – Retrieve the current window's handle.
- driver.getWindowHandles() – Retrieve handles for all open windows.

```
String mainWindow = driver.getWindowHandle(); // Get main window handle
for (String window : driver.getWindowHandles()) {
  driver.switchTo().window(window); // Switch to a new window
  if (!window.equals(mainWindow)) {
    driver.close(); // Close the new window
  }
}
driver.switchTo().window(mainWindow); // Return to the main window
```

iv. switchTo().activeElement()

Focuses on the element currently active or in focus on the webpage.

- **Purpose**: Switches the focus to the element that is currently active (focused) on the page.
- **Method**: driver.switchTo().activeElement()

```
// Get the active element and perform an action
 WebElement activeElement = driver.switchTo().activeElement();
System.out.println("Active element is: " + activeElement.getTagName());
activeElement.sendKeys("Testing active element");
```

v. driver.switchTo().newWindow(WindowType)

- **Purpose:** Automatically create and switch to a new tab or window in the browser.
  driver.switchTo().newWindow(WindowType.TAB); // Opens a new browser tab
  driver.switchTo().newWindow(WindowType.WINDOW); // Opens a new browser window

11. manage()

- **Declared in**: WebDriver interface
- **Implemented by**: RemoteWebDriver class
- To call this non-static method, create an object of WebDriver interface.
- **Argument**: None.
- **Returns**: Object of Options interface.
- **Purpose**: Useful to work with cookies, timeouts, browser window, and browser logs.

Detailed Sub-Methods and Examples

i. Cookies

Cookies are small pieces of data stored in the browser. This feature allows adding, deleting, or retrieving cookies during testing.

- **Purpose**: Handle browser cookies.
- **Sub-Methods**:
  - driver.manage().getCookies() – Retrieve all cookies.
  - driver.manage().getCookieNamed(String name) – Retrieve a specific cookie by its name.
  - driver.manage().addCookie(Cookie cookie) – Add a new cookie to the browser.
  - driver.manage().deleteCookie(Cookie cookie) – Delete a specific cookie.
  - driver.manage().deleteCookieNamed(String name) – Delete a cookie by name.
  - driver.manage().deleteAllCookies() – Clear all cookies from the browser.

ii. Timeouts

Timeouts are used to configure waits for browser operations like loading pages, finding elements, or script execution.

- **Purpose**: Define time limits for browser operations.
- **Sub-Methods**:
  - driver.manage().timeouts().implicitlyWait(Duration duration) – Set the time WebDriver waits when searching for elements.
  - driver.manage().timeouts().pageLoadTimeout(Duration duration) – Set the time limit for a page to load.
  - driver.manage().timeouts().scriptTimeout(Duration duration) – Set the time limit for script execution.

iii. Window Management

Window management helps control the browser window's state, size, and position.

- **Purpose**: Manage browser window behavior.
- **Sub-Methods**:

  - driver.manage().window().maximize() – Maximize the browser window.
  - driver.manage().window().minimize() – Minimize the browser window.
  - driver.manage().window().fullscreen() – Open the browser in full-screen mode.
  - driver.manage().window().getSize() – Retrieve the current size of the browser window.
  - driver.manage().window().getPosition() – Retrieve the current position of the browser window.
  - driver.manage().window().setSize(Dimension dimension) – Set a specific size for the browser window.
  - driver.manage().window().setPosition(Point point) – Set the position of the browser window.

iv. Logs *(Available in some browsers and for advanced debugging)*

Logs allow you to access browser or driver-level logs for debugging purposes.

- **Purpose**: Retrieve logs for debugging.
- **Sub-Methods**:
  - driver.manage().logs().get(String logType) – Retrieve logs of a specific type (e.g., "browser", "driver").
  - driver.manage().logs().getAvailableLogTypes() – Retrieve the list of available log types.

The driver.manage() method provides essential capabilities for:

1. Managing cookies for session control.
2. Configuring timeouts for smoother test execution.
3. Controlling browser window states and positions.
4. Debugging with logs to identify and fix issues.

12. findElement(By)

- **Declared in**: SearchContext interface
- **Implemented by**: RemoteWebDriver class
- The SearchContext interface is the parent of the WebDriver interface (inheritance).
- **Argument**: Takes a By class object.
- **Returns**: WebElement interface.
- **Purpose**: Useful to locate an element in the page source.
- **Note**: By is an abstract class, but it has 8 static methods (id, name, className, tagName, linkText, partialLinkText, xpath, cssSelector) to create its object.

Sub-Methods (for Locating Elements)

The By class in Selenium WebDriver is a fundamental utility used for locating elements on a web page. It is part of the org.openqa.selenium package and plays a crucial role in identifying web elements for interaction during test automation.

  - By is an **abstract class**, so you cannot create its object directly using new By().
  - Instead, it provides **static methods** to return By objects for different locator strategies (e.g., ID, name, XPath, etc.).

findElement is primarily used with various locator strategies to identify elements. Below are the common locator strategies used with findElement:

i. **By ID**
   - Locates an element using its unique id attribute.
   - **Syntax**: driver.findElement(By.id("element_id"))
   - **Use Case**: When the element has a unique id.

     Example:
     ```
     WebElement element = driver.findElement(By.id("username"));
     element.sendKeys("testUser");
     ```

ii. **By Name**
   - Locates an element using the name attribute.
   - **Syntax**: driver.findElement(By.name("element_name"))
   - **Use Case**: When the element has a name attribute.

     Example:
     ```
     WebElement element = driver.findElement(By.name("password"));
     element.sendKeys("password123");
     ```

iii. **By XPath**
   - Locates an element using its XPath expression.
   - **Syntax**: driver.findElement(By.xpath("xpath_expression"))
   - **Use Case**: For locating elements with complex conditions or relative paths.

     Example:
     ```
     WebElement element =
     driver.findElement(By.xpath("//input[@id='username']"));
     element.sendKeys("testUser");
     ```

iv. **By CSS Selector**
   - Locates an element using its CSS selector.
   - **Syntax**: driver.findElement(By.cssSelector("css_selector"))
   - **Use Case**: When elements have distinct CSS classes, IDs, or attributes.

     Example:
     ```
     WebElement element =
     driver.findElement(By.cssSelector("input#username"));
     element.sendKeys("testUser");
     ```

v. **By Class Name**
   - Locates an element using its CSS class name.
   - **Syntax**: driver.findElement(By.className("class_name"))
   - **Use Case**: When elements have a specific class.

     Example:
     ```
     WebElement element = driver.findElement(By.className("login-
     button"));
     element.click();
     ```

vi. By Tag Name

- Locates an element by its tag name.
- **Syntax**: driver.findElement(By.tagName("tag_name"))
- **Use Case**: When looking for elements like buttons, links, input fields, etc.

  Example:

  ```
  WebElement element = driver.findElement(By.tagName("button"));
  element.click();
  ```

vii. By Link Text

- Locates an anchor (<a>) element by its exact link text.
- **Syntax**: driver.findElement(By.linkText("link_text"))
- **Use Case**: When locating a link by its visible text.

  Example:

  ```
  WebElement element = driver.findElement(By.linkText("Click Here"));
  element.click();
  ```

viii. By Partial Link Text

- Locates an anchor (<a>) element by a partial match to its link text.
- **Syntax**: driver.findElement(By.partialLinkText("partial_text"))
- **Use Case**: When only part of the link text is known.

  Example:

  ```
  WebElement element = driver.findElement(By.partialLinkText("Click"));
  element.click();
  ```

13. findElements(By)

- **Declared in**: SearchContext interface
- **Implemented by**: RemoteWebDriver class
- The SearchContext interface is the parent of the WebDriver interface (inheritance).
- **Argument**: Takes a By class object.
- **Returns**: List<WebElement>
- **Purpose**: Useful to locate and collect one or more matched elements in the page source.

  ```
  List<WebElement> elements = driver.findElements(By.className("exampleClass"));
  ```