Selenium 4

Contents

Selenium 2

1. **Improvements** 2
2. **Selenium and Browser communication** 2
3. **Support of Relative Locators (Source – Dzone)** 2
4. **Window and Tab Management** 3
5. **No longer support for Opera and PhantomJS** 3
6. **ChromiumDriver and DevTools: (source from DZone)** 3
7. **Selenium 4 In a nutshell** 4
8. **Sources** 4

Selenium

1. Improvements

- Selenium IDE
  - This plug-in will provide support to Chrome also.
  - Allow to run selenium scripts on any browser
  - User can declare their own locator strategy
  - New CLI runner which is completely based on NodeJS
    - Parallel Execution - ability to execute the test case in parallel and provide details information (passed, failed, time taken for execution)
    - Run – this is complete based on WebDriver.
- Selenium Grid
  - Improved docker support
  - Setup
    - No longer to do any setup and start the hubs and nodes separately.
  - Improved UI
    - More user-friendly UI.
    - Display the relevant information about session running capacity… etc.,
    - Request tracing and logging with hooks will be improved.
    - Output logs in single line json format.
- Selenium Core
  - Improved the identification of parent frame -> Now we can directly switch to parent frame
    - driver.switchTo().parentFrame()
  - Few changes in WebDriver methods
    - (getPosition & getSize)– Replaced with – getRect
    - (setPosition & setSize)– Replaced with – setRect
    - Full screen and minimize method are added
    - Element screenshot is now possible.

2. Selenium and Browser communication

- W3C Capabilities -> Instead of JSON wire Protocol, to native support of W3C standard using session.
  - https://w3c.github.io/webdriver/

3. Support of Relative Locators (Source – Dzone)

This is a new feature of Selenium 4. Generally, when starting to write automation tests, you can find the element using by CSS or Xpath locators such as find by ID, name, link text, etc. Now with Selenium 4, you can find your element using Relative Locators. This means you can find your element that is close to other ones.

We have several methods to find a close-by (relative) element. These are:

- toLeftOf() : Element located to the left of specified element.
- toRightOf() : Element located to the right of the specified element.
- above() : Element located above with respect to the specified element.
- below() : Element located below with respect to the specified element.
- near() : Element is at most 50 pixels far away from the specified element. The pixel value can be modified.
- The methods which are represented above are overloaded accept a By or a WebElement. You can also see the usage of these properties in Figure 6:

```
WebElement mainHeading = driver.findElement(By.xpath("(//div[@class='Heading1'])"));
System.out.println("Main Heading: " + mainHeading.getText());

WebElement rightOfMainHeading = driver.findElement(withTagName("div")
        .toRightOf(By.xpath("(//div[@class='mainLabel'])")));
System.out.println("Right Of Main Heading: " + rightOfMainHeading.getText());

WebElement belowOfMainHeading = driver.findElement(withTagName("div")
        .below(By.id("ParagraphLabel")));
System.out.println("Below Of Main Heading: " + belowOfMainHeading.getText());
```

4. Window and Tab Management

- Running with different browsers In Same time.

  new ChromeDriver().switchTo().newWindow(WindowType.WINDOW);

- Running with different tab in one browser window In Same time.

  new ChromeDriver().switchTo().newWindow(WindowType.TAB);

5. No longer support for Opera and PhantomJS

- Well, since Opera is built using Chromium open source project (The project behind Chrome browser), the behavior of Chrome and Opera should be very similar. Hence, there are fewer chances of having different behaviors and you can simply test Chrome as the Opera browser. And you can still automate Chrome as it is still supported of course.
- And as for Headless Tests, you can use Chrome Headless or Firefox Headless instead of PhantomJS.

6. ChromiumDriver and DevTools: (source from DZone)

In Selenium 3, EdgeDriver and ChromeDriver have their own implementations and are directly inherited from the RemoteWebDriver shown in Figure 1 and Figure 2.

```
public class ChromeDriver extends RemoteWebDriver implements I
    private RemoteLocationContext locationContext;
    private RemoteWebStorage webStorage;
    private TouchScreen touchScreen;
    private RemoteNetworkConnection networkConnection;
```

```
public class EdgeDriver extends RemoteWebDriver {
    public EdgeDriver() {
        this(EdgeDriverService.createDefaultService(), new
    }
```

However, EdgeDriver and ChromeDriver extend from ChromiumDriver which has new ChromeDevTools implementations, and you can see the new methods in Figure 3 and Figure 4;

```
public class EdgeDriver extends ChromiumDriver {
    public static final String DRIVER_USE_EDGE_EDGEHTML = "webdriver.edge.edgehtml";
```

EdgeDriver methods

```
public class ChromeDriver extends ChromiumDriver {
    public ChromeDriver() {
```

When you open the Chrome browser, you can inspect the Chrome developer tools. Now, as you can see in the screenshot below, you can access these developer tools by calling the methods in Selenium 4.

```
public class ChromiumDriver extends RemoteWebDriver implements HasDevTools, HasTouchScreen, LocationCo
    private final RemoteLocationContext locationContext = new RemoteLocationContext(this.getExecuteMet
    public Map<String, Object> executeCdpCommand(String commandName, Map<String, Object> parameters)
    public DevTools getDevTools() {...}
    public String getCastSinks() {...}
    public String getCastIssueMessage() {...}
    public void selectCastSink(String deviceName) {...}
    public void startTabMirroring(String deviceName) {...}
    public void stopCasting(String deviceName) {...}
```

7. Selenium 4 In a nutshell
8. Sources

- https://blog.testproject.io/2019/05/01/open-source-selenium-4-test-automation/
- https://cloudqa.io/selenium-4-releasing-soon-2018/
- https://dzone.com/articles/new-comings-with-seleniumnbsp4