Ravindra Gilakaa (7989870220)

SELENIUM NOTES

1. Overview of Selenium (Selenium Suite Components:)

1. Selenium IDE (Integrated Development Environment)

- Think of it like a video recorder for your browser actions.
- You click, type, scroll on a website → Selenium IDE records it → you can replay it.
- It's mainly for quick testing and learning, not for serious automation projects.

Example: Like recording a cooking tutorial — later you can press play to watch the steps happen again.

2. Selenium RC (Remote Control) – Old & Retired

- This was the first big version of Selenium for writing code in multiple programming languages.
- But it was slow and needed a separate server running.

Example: Like an old phone that works but is slow — now replaced by smartphones (WebDriver).

3. Selenium WebDriver (The Star Player)

- This is what most people mean when they say "Selenium" today.
- Let's you write code in Java, Python, C#, JavaScript, etc. to control the browser directly.
- No separate server needed — faster and more powerful than RC.

Example: Like driving your own car instead of taking a bus — you're in direct control of where you go and how fast.

Real-world use: Automating Google search, filling forms, scraping data, testing login pages, etc.

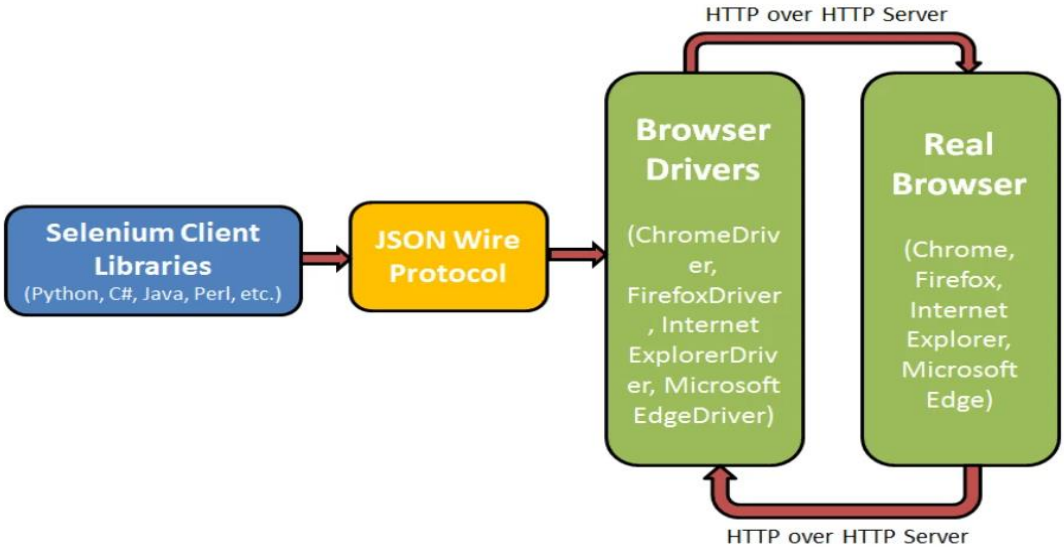
4. Selenium Grid (Teamwork for Browsers)

- Let's run tests on multiple computers & browsers at the same time.
- Good for parallel testing and testing on different browser/OS combinations without owning them all.

Example: Like having multiple chefs cook the same dish in different kitchens — you get results faster**.**
Real-world use: You can test your website on Chrome in Windows, Firefox in Mac, and Edge in Linux — all at the same time.

2. Selenium WebDriver Basics

1. WebDriver architecture



Selenium Client Libraries (blue box)

- These are the special tools for different programming languages (like Java, Python, C#) that let you tell the browser what to do.
- Think of it like **the remote control for your browser**.

JSON Wire Protocol (yellow box)

- This is like a translator.
- It changes your code into a special format (JSON) so it can be understood by the browser drivers.

Browser Drivers (green box, left)

- Each browser has its own driver (Chrome → ChromeDriver, Firefox → GeckoDriver, Edge → EdgeDriver, etc.).
- The driver receives the instructions from JSON Wire Protocol and tells the browser what to do.

Real Browser (green box, right)

- This is the actual web browser (Chrome, Firefox, Edge, etc.) where your script actions are performed (clicks, typing, navigation).

Flow:

You → write script in Java/Python → Selenium library sends it via JSON Wire

Protocol → Browser Driver understands and talks to the browser → Browser performs the action → Result comes back the same way.

2. Driver executables & path setup

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

2. Firefox
   System.setProperty("webdriver.gecko.driver", "C:\\path\\to\\geckodriver.exe");

3. Edge
   System.setProperty("webdriver.edge.driver", "C:\\path\\to\\msedgedriver.exe");

4. Internet Explorer
   System.setProperty("webdriver.ie.driver", "C:\\path\\to\\IEDriverServer.exe");

5. Safari (Mac only, no path setup needed if Safari Driver is enabled in Safari's Develop menu)
   System.setProperty("webdriver.safari.driver", "/usr/bin/safaridriver");

3. Headless browser

```
ChromeOptions options = new ChromeOptions();
options.addArguments("--headless"); // Run in headless mode
options.addArguments("--disable-gpu"); // Optional, disable GPU rendering
```

3.Locators in Selenium

1. By CSS ID:
2. By CSS class name:
3. By name attribute:
4. By DOM structure or Xpath:
5. by tagName:
6. By link text:
7. By partial link text:
8. By HTML tag name:

findElement vs findElements

Feature	findElement	findElements
What it does	Finds **one element** on the web page	Finds **all matching elements** on the web page
Return type	Single WebElement	List of WebElements

If element not found	Throws **NoSuchElementException**	Returns an **empty list**
Usage example	driver.findElement(By.id("login")) → returns the login button	driver.findElements(By.className("menu")) → returns a list of all menu items
Best for	When you want **only one element**	When you want **multiple elements**

Absolute vs Relative XPath

Point	Absolute XPath	Relative XPath
Start Symbol	`/`	`//`
Meaning	Full path from the first tag (<html>) to your element	Short path from anywhere in the page
Flexibility	Not flexible – breaks if page layout changes	Flexible – works even if layout changes
Speed	Fast	Slightly slower
When to Use	When the page structure is fixed	When structure can change
Example	`/html/body/div[1]/div[2]/form/input[1]`	`//input[@id='username']`

CSS VS Xpath:

Point	CSS Selector	XPath
Syntax	Uses CSS style syntax (e.g., tag#id, tag.class)	Uses path syntax with / or //
Readability	Easy to read and write	More complex
Performance	Faster in most browsers	Slightly slower
Flexibility	Good for style-related selections	More powerful – can navigate both forward and backward in DOM
Attribute Selection	tag[attribute='value']	//tag[@attribute='value']
Text Search	Not supported directly	//tag[text()='value']

When to Use	When you need speed and simplicity	When you need complex DOM traversal

get() vs navigate()

Feature	get()	navigate()
Purpose	Opens a web page	Can open a page **and** move forward/back/refresh
Syntax example	driver.get("https://example.com")	driver.navigate().to("https://example.com")
Can do back/forward	No	Yes (back(), forward())
Can refresh page	No	Yes (refresh())
Best for	Just opening a web page	Opening a page and also moving around in browser history

4.Commands

1.Web Element Operations

- WebElement element;
- element.clear();
- element.sendKeys("text");
- element.getText();
- element.getAttribute("attributeName");
- element.isDisplayed();
- element.isEnabled();
- element.isSelected();
- element.submit();

2.Driver Operations

- driver.getTitle();
- driver.getPageSource();
- driver.getCurrentUrl();
- driver.quit();
- driver.close();

3.Navigation Commands

- driver.get("http://example.com");
- driver.navigate().to("http://example.com");
- driver.navigate().back();
- driver.navigate().forward();
- driver.navigate().refresh();

5.Frames

- driver.switchTo().frame(int index);
- driver.switchTo().frame(String nameOrId);
- driver.switchTo().frame(WebElement frameElement);
- driver.switchTo().defaultContent();

6.Alerts

- driver.switchTo().alert().getText();
- driver.switchTo().alert().accept();
- driver.switchTo().alert().dismiss();
- driver.switchTo().alert().sendKeys("text");

7. Waits in Selenium

Selenium 3 - Implicit Wait

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

Explicit Wait

```
WebDriverWait wait = new WebDriverWait(driver, 10); // 10 seconds
 WebElement element1 = wait.until(
 ExpectedConditions.visibilityOfElementLocated(By.id("username")));
```

Fluent Wait

```
Wait<WebDriver> fluentWait = new FluentWait<WebDriver>(driver)
.withTimeout(20, TimeUnit.SECONDS)       // Total wait time
.pollingEvery(2, TimeUnit.SECONDS)       // Polling frequency
.ignoring(NoSuchElementException.class); // Ignore exception
```

```
        WebElement element2 = fluentWait.until(new Function<WebDriver,
WebElement>() {
        public WebElement apply(WebDriver driver) {
        return driver.findElement(By.id("username"));
                }
        });
```

8.Dropdowns (Select class)

```
Select select = new Select(dropdown);
select.selectByVisibleText("India");
select.selectByValue("IN");
select.selectByIndex(2);

if (select.isMultiple()){
select.selectByVisibleText("Java");
select.selectByVisibleText("Selenium");

select.deselectByVisibleText("Java");
select.deselectAll();
}

List<WebElement> options = select.getOptio
```

8.Radio Buttons

```
if (!radio.isSelected()) {
   radio.click();}
```

9.Mouse actions

```
Actions actions = new Actions(driver);
```

1. Hover over elemen

```
WebElement menu = driver.findElement(By.id("menu"));
actions.moveToElement(menu).perform();
Thread.sleep(1000);
```

2. Drag & Drop

```
WebElement source = driver.findElement(By.id("draggable"));
WebElement target = driver.findElement(By.id("droppable"));
actions.dragAndDrop(source, target).perform();
```

```
Thread.sleep(1000);
```

3. Double Click

```
WebElement doubleClickBtn = driver.findElement(By.id("doubleClickBtn"));
actions.doubleClick(doubleClickBtn).perform();
Thread.sleep(1000);
```

4. Right Click

```
WebElement rightClickBtn = driver.findElement(By.id("rightClickBtn"));
actions.contextClick(rightClickBtn).perform();
Thread.sleep(1000);
```

5.Move To element

```
actions.moveToElement(menu).perform();
```

6.Keyboard Actions

Example: Type in uppercase using SHIFT key

```
WebElement inputBox = driver.findElement(By.id("searchBox"));
actions.keyDown(Keys.SHIFT).sendKeys(inputBox,
"selenium").keyUp(Keys.SHIFT).perform();
Thread.sleep(1000);
```

Example: Select all text (CTRL + A) and delete

```
actions.keyDown(Keys.CONTROL).sendKeys("a").keyUp(Keys.CONTROL).send
Keys(Keys.DELETE).perform();
driver.quit();
}}
```

7.Difference Between build() and perform() in Selenium Actions Class

1. perform()

- Meaning: Executes the action immediately.
- When to use: When you have only one action to perform.

Example:

```
Actions actions = new Actions(driver);
WebElement element = driver.findElement(By.id("menu"));
actions.moveToElement(element).perform();
```

Explanation: No need to combine actions, it runs the hover immediately.

2. build()

- Meaning: Combines multiple actions into one sequence and prepares them.
- When to use: When you have multiple actions to perform together.
- Note: build() only prepares; you must still call perform() to execute.

Example:

```
Actions actions = new Actions(driver);
WebElement source = driver.findElement(By.id("drag"));
WebElement target = driver.findElement(By.id("drop"));
actions.moveToElement(source)
.clickAndHold()
.moveToElement(target)
.release()
.build()
.perform();
```

Explanation: build() joins multiple steps into one action chain, perform() executes them.

Easy way to remember:
perform() = Do it now
build() = Prepare it first, then do it.

11.Switch to window

1. Get current window handle
```
driver.getWindowHandle();
```

2. Get all window handles
```
driver.getWindowHandles();
```

3. Switch to a specific window
```
driver.switchTo().window(windowHandle);
```

4. Switch back to parent/default window
```
driver.switchTo().defaultContent();
```

5. Close the current window
```
driver.close();
```

12. Taking Screenshots

Full Page Screenshot

```
File srcFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
```

Element Screenshot (Only selenium - 4)

```
File srcFile = element.getScreenshotAs(OutputType.FILE);
```

13. Java script Executor

```
JavascriptExecutor js = (JavascriptExecutor) driver;
WebElement element = driver.findElement(By.id("myElement"));
```

1. JavaScript Click

```
js.executeScript("arguments[0].click();", element);
```

2. Move to element (scroll into view)

```
js.executeScript("arguments[0].scrollIntoView(true);", element);
```

3. Scroll page by pixels (down 500px)

```
js.executeScript("window.scrollBy(0,500);");
```

4. Send text to element

```
js.executeScript("arguments[0].value='Hello World';", element);
```

14. Get and Set element position

1.Get element position (X and Y coordinates)

```
Point point = element.getLocation();
int xPosition = point.getX();
int yPosition = point.getY();
System.out.println("Element X Position: " + xPosition);
System.out.println("Element Y Position: " + yPosition);
```

2. Get element size (Height and Width)

```
Dimension size = element.getSize();
int height = size.getHeight();
int width = size.getWidth();
System.out.println("Element Height: " + height);
System.out.println("Element Width: " + width);
```

3.Get both position and size together (getRect())

```
Rectangle rect = element.getRect();
```

```
System.out.println("Rect X: " + rect.getX());
System.out.println("Rect Y: " + rect.getY());
System.out.println("Rect Height: " + rect.getHeight());
System.out.println("Rect Width: " + rect.getWidth());
```

Set Position

```
Point point = new Point(300, 200); // Create a Point object with x=300, y=200
driver.manage().window().setPosition(point);
```

15. Xpaths

Xpaths Types

```
Xpath=//*[contains(@type,'sub')]
Xpath=//label[starts-with(@id,'message')]
Xpath=//input[contains(text(),'text')]
Xpath=//td[text()='UserID']
```

XPath Axis	Meaning
following	Selects all nodes after the current node — no matter the level or type
following-sibling	Selects only the siblings (same level) that come after the current node.

ancestor

The ancestor axis selects all parent, grandparent, etc. elements of the current node — upward in the DOM tree.

```
//input[@id='username']/ancestor::div
```

Chaild

Selects all children elements of the current node

```
//div[@class='container']//child::label
```

parent

Selects the parent of the current node

```
//input[@id='username']/parent::form
```

self

The self axis refers to the current node itself — not its parent, not its child — just that exact element.

Xpath =//*[@type='password']//self::input

It’s often used with conditions or functions like self::node(), self::div, etc., mainly in filtering or refining selections.

descendant

Selects the descendants of the current node
//div[@class='container']/descendant::label

Advanced concepts

Desired Capabilities

1. What is DesiredCapabilities?

• It is used to set browser properties before starting the browser.
• You can define:
--> Browser name (Chrome, IE, Firefox)
--> Browser version
--> Platform/OS (Windows, Mac, Linux)
--> Other settings like device name, app path, etc.

2. Common Methods

Method	What it does	Example
setBrowserName(String name)	Set browser name	capabilities.setBrowserName("chrome");
getBrowserName()	Get browser name	capabilities.getBrowserName();
setVersion(String version)	Set browser version	capabilities.setVersion("116.0");
getVersion()	Get browser version	capabilities.getVersion();

setPlatform(Platform platform)	Set OS	capabilities.setPlatform(Platform.WINDOWS);
getPlatform()	Get OS	capabilities.getPlatform();
setCapability(String key, Object value)	Set any capability	capabilities.setCapability("ignoreZoomSetting", true);
getCapability(String key)	Get capability value	capabilities.getCapability("ignoreZoomSetting");

Code

```
DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setBrowserName("internet explorer");
capabilities.setPlatform(Platform.WINDOWS);
capabilities.setVersion("11");
WebDriver driver = new InternetExplorerDriver(capabilities);
```

4. Key Points to Remember

1. DesiredCapabilities = Browser settings before running tests.
2. setCapability() = Set any property.
3. getCapability() = Check current property.
4. Useful for cross-browser and cross-platform testing.

Handling SSL Certificates

1.Chrome – Accept SSL Certificates

```
DesiredCapabilities capabilities = DesiredCapabilities.chrome();
capabilities.setAcceptInsecureCerts(true); // Accept SSL certificate

ChromeOptions options = new ChromeOptions();
options.merge(capabilities);
WebDriver driver = new ChromedRiver(options);
```

2.Firefox – Accept SSL Certificates

```
DesiredCapabilities capabilities = DesiredCapabilities.firefox();
capabilities.setAcceptInsecureCerts(true); // Accept SSL certificate

FirefoxOptions options = new FirefoxOptions();
options.merge(capabilities);

WebDriver driver = new FirefoxDriver(options);
```

Handling Auto Suggestions

```
WebElement searchBox = driver.findElement(By.name("q"));
searchBox.sendKeys("selenium");

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
List<WebElement> suggestions = wait.until(
ExpectedConditions.visibilityOfAllElementsLocatedBy(
By.xpath("//ul[@role='listbox']//li//span")));

for (WebElement suggestion : suggestions) {
if (suggestion.getText().equalsIgnoreCase("selenium tutorial")) {
        suggestion.click();
break;
}
}
```

Handling Tool Trip

```
driver.get("https://www.selenium.dev/documentation/webdriver/elements/");

Find element with tooltip
WebElement element = driver.findElement(By.linkText("WebDriver"));

Get tooltip text from 'title' attribute
String tooltipText = element.getAttribute("title");
System.out.println("Tooltip text: " + tooltipText);
```

Broken Links

```
List<WebElement> links = driver.findElements(By.tagName("a"));

System.out.println("Total links found: " + links.size());

for (WebElement linkElement : links) {

String url = linkElement.getAttribute("href");


// Skip if href is null or empty

if (url == null || url.isEmpty()) {
```

```
System.out.println("Empty or null link skipped");
continue;
}
try {
URL link = new URL(url);
URLConnection urlConnection = link.openConnection();
HttpURLConnection connection = (HttpURLConnection) urlConnection;

// Use HEAD request to get response code
connection.setRequestMethod("HEAD");
connection.connect();
int responseCode = connection.getResponseCode();
if (responseCode >= 400) {
System.out.println("Broken link: " + url + " --> " + responseCode);
} else {
System.out.println("Valid link: " + url + " --> " + responseCode);
}
} catch (Exception e) {
System.out.println("Exception for link: " + url + " --> " + e.getMessage());
}
}
```

File Verification after downloading

```
String downloadFolder = "C:\\Users\\YourUser\\Downloads";
// Get all files in the folder
File folder = new File(downloadFolder);
File[] listOfFiles = folder.listFiles(); // returns an array of files

// Name of the file we want to verify
String expectedFile = "sample.txt";
```

```
boolean fileFound = false;


// Loop through the files
for (File file : listOfFiles) {
if (file.isFile() && file.getName().equals(expectedFile)) {
fileFound = true;
break;
}
}

// Print result
if (fileFound) {
System.out.println("File downloaded successfully: " + expectedFile);
} else {
System.out.println("File not found: " + expectedFile);
}
}
}
```

Auto It code

Auto It code for upload code

```
; Wait for the File Upload window
WinWait("Open")

; Set the file path in the File Name text box
ControlSetText("Open", "", "Edit1",
"C:\\Users\\YourUser\\Documents\\sample.txt")

; Click the Open button
ControlClick("Open", "", "Button1")
```

Save as ---------→ FileUpload.exe

```
WebDriver driver = new ChromeDriver();
driver.get("https://example.com/upload"); // Replace with your URL
```

```
// Click the button that opens file upload dialog
driver.findElement(By.id("uploadBtn")).click();

// Run the compiled AutoIt EXE to handle the Windows file dialog
Runtime.getRuntime().exec("C:\\Users\\YourUser\\AutoItScripts\\FileUpload.exe")
;
```

Auto It code for handling alerts

```
; Wait for the alert window to appear
WinWait("Alert Title")  ; Replace "Alert Title" with the actual window title

; Activate the alert window
WinActivate("Alert Title")

; Click the OK button (usually Button1)
ControlClick("Alert Title", "", "Button1")

; Optional: You can also click Cancel if needed
; ControlClick("Alert Title", "", "Button2")
```

Robot Class

```
Robot robot = new Robot();
robot.keyPress(KeyEvent.VK_ENTER);
robot.keyRelease(KeyEvent.VK_ENTER);

// Press Tab key
robot.keyPress(KeyEvent.VK_TAB);
robot.keyRelease(KeyEvent.VK_TAB);

// Press Escape key
robot.keyPress(KeyEvent.VK_ESCAPE);
robot.keyRelease(KeyEvent.VK_ESCAPE);

// Press Ctrl + S (example for save)
robot.keyPress(KeyEvent.VK_CONTROL);
robot.keyPress(KeyEvent.VK_S);
robot.keyRelease(KeyEvent.VK_S);
robot.keyRelease(KeyEvent.VK_CONTROL);
```

Handling Ajax calls and Alerts

1. What is AJAX?

- AJAX updates part of a webpage without reloading the whole page.

Example: You click "Search" and results appear after a few seconds.

- Problem: Selenium may try to find elements too early and fail.
- Remember: AJAX = updates page in the background.

2. Handling AJAX in Selenium

- Do not use Thread.sleep() (not reliable).
- Use Explicit Wait (WebDriverWait) to wait until element is ready.

Example:

```
// Click button that triggers AJAX
driver.findElement(By.id("searchButton")).click();

// Wait for result to appear
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
WebElement result = wait.until(
    ExpectedConditions.visibilityOfElementLocated(By.id("resultTable"))
);

System.out.println(result.getText());
```

visibilityOfElementLocated = wait until element is visible
presenceOfElementLocated = wait until element exists in HTML

3. Handling AJAX alerts

Sometimes AJAX triggers pop-up alerts. Use Alert in Selenium.

Example:

```
// Wait until alert is present
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
Alert alert = wait.until(ExpectedConditions.alertIsPresent());

// Read alert message and accept
System.out.println(alert.getText());
alert.accept(); // click OK
alert.dismiss(); // click Cancel if needed
```

Remember:
Always wait for the alert.
Use accept() for OK, dismiss() for Cancel.

Ravindra Gilakaa (7989870220)

Basic and Critical Selenium challenges

Basic Challenges:

1. Limited and complex cross-browser testing setup
2. Difficulty in synchronizing events with dynamic web pages
3. Poor handling of pop-ups and alerts
4. Challenges with dynamic web content and frequently changing elements
5. Limited built-in reporting and analytics
6. Slow test execution, especially on large test suites
7. Difficulty in reliably handling dynamic web elements
8. Minimal error handling and recovery mechanisms
9. Restricted to testing on desktop browsers only
10. Multi-language support challenges
11. Platform independence limitations
12. Integration capabilities with other tools
13. Extensibility limitations
14. Automated UI testing complexity
15. Timeout handling and event synchronization challenges

Critical Challenges:

1. Frequent false positive and false negative test results
2. Lack of native support for mobile application testing
3. Inability to automate CAPTCHA and OTP workflows
4. High maintenance overhead due to frequent UI changes
5. No support for Windows-based desktop applications
6. Limited support for native mobile applications without third-party tools
7. Testing for multiple mobile operating systems
8. Scalability issues

Exceptions in Selenium

Basic Exceptions in Selenium:

1. NoSuchElementException – Element not found on the page
2. ElementNotVisibleException – Element is present but not visible
3. ElementNotSelectableException – Element cannot be selected
4. TimeoutException – Waited too long for an element or condition
5. StaleElementReferenceException – Element is no longer attached to the DOM
6. InvalidElementStateException – Cannot interact with the element

7. WebDriverException – Generic WebDriver error
8. NoSuchFrameException – Frame not found
9. NoSuchWindowException – Window not found
10. InvalidSelectorException – Wrong locator or selector syntax

Critical Exceptions in Selenium:

1. SessionNotCreatedException – Browser session could not be created
2. ElementClickInterceptedException – Cannot click element due to overlay
3. MoveTargetOutOfBoundsException – Cannot move mouse to element
4. JavascriptException – JavaScript execution error
5. UnexpectedAlertPresentException – Unexpected alert blocks execution
6. TimeoutException (critical scenarios) – Critical waits failing
7. WebDriverException (browser crash) – Browser crashed or stopped responding
8. InvalidCookieDomainException – Wrong domain for cookie operations

Chrome, Firefox Profile and PROXY setup

Firefox Profile:

```
System.setProperty("webdriver.firefox.bin", "path/to/firefox/binary");
FirefoxProfile fp = new FirefoxProfile();
File file = new File("path/to/extension.xpi");
fp.addExtension(file);
DesiredCapabilities dc = DesiredCapabilities.firefox();
dc.setCapability(FirefoxDriver.PROFILE, fp);
WebDriver driver = new RemoteWebDriver(dc);
```

Chrome Options:

```
System.setProperty("webdriver.chrome.driver", "path/to/chromedriver.exe");
ChromeOptions options = new ChromeOptions();
File extFile = new File("path/to/extension.crx");
options.addExtensions(extFile);
DesiredCapabilities dc = DesiredCapabilities.chrome();
dc.setCapability(ChromeOptions.CAPABILITY, options);
WebDriver driver = new RemoteWebDriver(dc);
```

Proxy:

```
String PROXY = "localhost:8080";
Proxy proxy = new Proxy();
proxy.setHttpProxy(PROXY).setFtpProxy(PROXY).setSslProxy(PROXY);
DesiredCapabilities cap = new DesiredCapabilities();
cap.setCapability(CapabilityType.PROXY, proxy);
WebDriver driver = new FirefoxDriver(cap);
```

```
WebDriver driver = new ChromeDriver(cap);
```

Selenium WebDriver Family

1. SearchContext (Interface)

Root interface. The base for WebDriver.

Methods:
- WebElement findElement(By locator)
- List<WebElement> findElements(By locator)

(All drivers inherit these two basic methods.)

2. WebDriver (Interface, extends SearchContext)

Parent interface that defines browser-level operations.

Methods:
- void get(String url)
- String getCurrentUrl()
- String getTitle()
- List<WebElement> findElements(By locator) [from SearchContext]
- WebElement findElement(By locator) [from SearchContext]
- String getPageSource()
- void close()
- void quit()
- Set<String> getWindowHandles()
- String getWindowHandle()
- TargetLocator switchTo()
- Navigation navigate()
- Options manage()

3. RemoteWebDriver (Class, implements WebDriver)

Base class that actually implements WebDriver interface.
Most browser drivers extend this.

Important Methods:
- execute(String command, Map<String, ?> parameters)
- get(String url) [implemented]
- findElement(By locator) [implemented]
- findElements(By locator)
- getTitle()
- getCurrentUrl()
- close()

- quit()
- getWindowHandle()
- getWindowHandles()
- manage()
- navigate()
- switchTo()

Extra Interfaces implemented:
- JavascriptExecutor (executeScript, executeAsyncScript)
- TakesScreenshot (getScreenshotAs)
- HasCapabilities (getCapabilities)

4. Browser Drivers (extend RemoteWebDriver)

These are concrete classes for each browser.

a) ChromeDriver
- Extends RemoteWebDriver
- Launches and controls Chrome browser.
- Inherits all RemoteWebDriver methods.

b) FirefoxDriver
- Extends RemoteWebDriver
- For Firefox browser.

c) EdgeDriver
- Extends RemoteWebDriver
- For Microsoft Edge browser.

d) SafariDriver
- Extends RemoteWebDriver
- For Safari browser.

e) InternetExplorerDriver
- Extends RemoteWebDriver
- For Internet Explorer browser.

(All above drivers just call the browser-specific executables,
but methods are same because they inherit from RemoteWebDriver.)

5. Other Important Interfaces

These are often implemented by RemoteWebDriver:

a) JavascriptExecutor
- Object executeScript(String script, Object... args)
- Object executeAsyncScript(String script, Object... args)

b) TakesScreenshot
- <X> X getScreenshotAs(OutputType<X> target)

c) HasCapabilities
- Capabilities getCapabilities()

d) WebStorage (only for some drivers)
- LocalStorage getLocalStorage()
- SessionStorage getSessionStorage()

SearchContext (Interface)
└── WebDriver (Interface, extends SearchContext)
└── RemoteWebDriver (Class, implements WebDriver + extra interfaces)
├── ChromeDriver (Class, extends RemoteWebDriver)
├── FirefoxDriver (Class, extends RemoteWebDriver)
├── EdgeDriver (Class, extends RemoteWebDriver)
├── SafariDriver (Class, extends RemoteWebDriver)
└── InternetExplorerDriver (Class, extends RemoteWebDriver)

SELENIUM COOKIES

What are Cookies?

- Small pieces of data stored in the browser.
- Websites use cookies to remember:
  * Login information
  * User preferences (dark/light mode)
  * Session details

In Selenium, we can get, add, delete, and manage cookies for automation testing.

Main Cookie Operations

1. Get all cookies → Retrieve all cookies stored by the browser.
2. Get a specific cookie → Retrieve a cookie by its name.
3. Add a cookie → Insert a new cookie into the browser.
4. Delete a specific cookie → Remove a cookie by name.
5. Delete all cookies → Clear the entire cookie storage.

Example Code (Java – Selenium 4)

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.Cookie;
import java.util.Set;

public class CookiesExample {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();

        // Open website
        driver.get("https://www.flipkart.com");

        // 1. Get all cookies
        Set<Cookie> cookies = driver.manage().getCookies();
        System.out.println("Total Cookies: " + cookies.size());
        for (Cookie c : cookies) {
            System.out.println(c.getName() + " : " + c.getValue());
        }

        // 2. Get a specific cookie by name
        Cookie sessionCookie = driver.manage().getCookieNamed("session-id");
        if (sessionCookie != null) {
            System.out.println("Session Cookie: " + sessionCookie.getValue());
        }

        // 3. Add a new cookie
        Cookie newCookie = new Cookie("myCookie", "12345");
        driver.manage().addCookie(newCookie);
        System.out.println("Cookie added: " +
driver.manage().getCookieNamed("myCookie"));

        // 4. Delete a specific cookie
        driver.manage().deleteCookieNamed("myCookie");
        System.out.println("Cookie 'myCookie' deleted.");

        // 5. Delete all cookies
        driver.manage().deleteAllCookies();
        System.out.println("All cookies deleted.");

        driver.quit();
    }
}
```

Example Output

Total Cookies: 5
session-id : abc123
user-preferences : dark-mode

Cookie added: myCookie=12345
Cookie 'myCookie' deleted.
All cookies deleted.

Why Use Cookies in Testing?

- Skip login → Save login cookies once and reuse them.
- Session testing → Validate behavior when session cookies expire.
- User preferences → Test features like theme (dark/light mode).

🤝

Let's keep learning and growing together — feel free to share your thoughts or favorite Selenium tips! 💬