Key Concepts in Selenium

By Rupendra Ragala

Selenium

WebDriver

- Definition: A Selenium API that directly communicates with the browser to automate tasks.

Example:

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

Selenium Grid

- Definition: Used for parallel execution on multiple machines or browsers.
- Example: Set up a Selenium Grid hub and connect nodes to run tests simultaneously on Chrome and Firefox.

Selenium IDE

- Definition: A browser-based tool for record-and-playback testing.
- Example: Record a login flow in Selenium IDE and export the test as WebDriver code.

Selenium

DesiredCapabilities

- Definition: Allows setting browser-specific properties like platform and version.
- Example:

```
DesiredCapabilities caps = new DesiredCapabilities();
caps.setBrowserName("chrome");
caps.setPlatform(Platform.WINDOWS);
```

RemoteWebDriver

- Definition: Executes tests on remote machines or in a Selenium Grid setup.
- Example:

```
WebDriver driver = new RemoteWebDriver(new URL("http://localhost:4444"), caps);
```

Locators

- Definition: Strategies to find elements on a webpage (e.g., ID, Name, XPath, CSS Selector).
- Example:

```
WebElement element = driver.findElement(By.id("username"));
```

Selenium

WebElement

- Definition: Represents elements on a webpage, like buttons or text fields.
- Example:

```
WebElement button = driver.findElement(By.name("submit"));
button.click();
```

Implicit Wait

- Definition: Sets a default waiting time for all elements.
- Example:

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

Explicit Wait

- Definition: Waits for a specific condition to be true before proceeding.
- Example:

```
WebDriverWait wait = new WebDriverWait(driver, 20);
wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("login")));
```

Selenium

Fluent Wait

- Definition: Allows setting polling intervals and ignoring exceptions during waiting.
- Example:

```
Wait<WebDriver> wait = new FluentWait<>(driver)
                        .withTimeout(Duration.ofSeconds(30))
                        .pollingEvery(Duration.ofSeconds(5))
                        .ignoring(NoSuchElementException.class);
```

Actions Class

- Definition: Used for advanced user interactions like drag-and-drop, hovering, etc.
- Example:

```
Actions actions = new Actions(driver);
actions.moveToElement(element).click().build().perform();
```

JavaScriptExecutor

- Definition: Executes JavaScript code within the browser.
- Example:

```
JavascriptExecutor js = (JavascriptExecutor) driver;
js.executeScript("document.getElementById('username').value='admin';");
```

Selenium

Window Handles

- Definition: Manages multiple browser windows or tabs.
- Example:

```
for (String handle : driver.getWindowHandles()) {
    driver.switchTo().window(handle);
}
```

Frames and IFrames

- Definition: Allows switching to specific sections of a webpage within a frame.
- Example:

```
driver.switchTo().frame("frameName");
```

Alerts

- Definition: Handles JavaScript alerts or pop-ups.
- Example:

```
Alert alert = driver.switchTo().alert();
alert.accept();
```

Selenium

TestNG

- Definition: A testing framework for running and organizing Selenium tests.
- Example: Annotate test cases using @Test in TestNG.

Maven

- Definition: A build automation tool for managing project dependencies.
- Example: Add Selenium WebDriver dependency to pom.xml.

Jenkins

- Definition: A CI/CD tool for running Selenium tests in a pipeline.
- Example: Set up a Jenkins job to trigger Selenium tests after a code push.

Selenium

Handling File Uploads

- Definition: Automates file uploads by interacting with input tags.
- Example:

```
WebElement upload = driver.findElement(By.id("fileUpload"));
upload.sendKeys("C:\\path\\to\\file.txt");
```

Downloading Files

- Definition: Handles file downloads by setting browser preferences.
- Example (for Chrome):

```
ChromeOptions options = new ChromeOptions();
options.addArguments("download.default_directory=C:\\Downloads");
WebDriver driver = new ChromeDriver(options);
```

Handling Dynamic Web Elements

- Definition: Automates elements that change dynamically using XPath or CSS.
- Example:

```
driver.findElement(By.xpath("//button[contains(text(),'Start')]")).click();
```

Selenium

Shadow DOM

- Definition: Automates elements within a shadow root (hidden DOM).
- Example:

```
JavascriptExecutor js = (JavascriptExecutor) driver;
WebElement shadowRoot = (WebElement) js.executeScript("return document
    .querySelector('shadow-host').shadowRoot");
```

Automating Captchas

- Definition: Captchas are difficult to automate but can use third-party tools like OCR or APIs.
- Example: Use services like 2Captcha to solve captchas programmatically.

Parallel Test Execution

- Definition: Runs multiple tests simultaneously using TestNG or Selenium Grid.
- Example: Configure testng.xml to specify parallel execution:

```
<suite name="Parallel Test Suite" parallel="tests">
    <test name="Test1">...</test>
    <test name="Test2">...</test>
</suite>
```

Selenium

Cross-Browser Testing

- Definition: Ensures compatibility across multiple browsers (Chrome, Firefox, Edge).
- Example:

```
WebDriver driver = new FirefoxDriver(); // or ChromeDriver, EdgeDriver
```

Mobile Browser Automation

- Definition: Automates mobile browsers using tools like Appium with Selenium.
- Example: Use Appium to automate Chrome on Android devices.

Handling HTTPS Certificates

- Definition: Handles untrusted certificates using browser capabilities.
- Example:

```
ChromeOptions options = new ChromeOptions();
options.setAcceptInsecureCerts(true);
```

Selenium

Selenium Docker Integration

- Definition: Runs Selenium in Docker containers for scalability and CI/CD pipelines.
- Example: Use the official Selenium Docker images like selenium/standalone-chrome

Data-Driven Testing (DDT)

- Definition: Uses external data (Excel, JSON, DB) for test inputs.
- Example: Read data from an Excel sheet using Apache POI.

```
FileInputStream file = new FileInputStream(new File("data.xlsx"));
Workbook workbook = new XSSFWorkbook(file);
```

Keyword-Driven Testing

- Definition: Separates test scripts from keywords to improve reusability.
- Example: Keywords like click(), sendKeys() can be mapped to methods.

Selenium

BDD with Cucumber

- Definition: Combines Selenium with Cucumber for behavior-driven testing.
- Example: gherkin

```
Feature: Login functionality
  Scenario: Successful login
    Given I am on the login page
    When I enter valid credentials
    Then I should see the dashboard
```

Debugging Selenium Tests

- Definition: Use breakpoints and logs to identify issues.
- Example: Use IDEs like IntelliJ or Eclipse to debug step by step.

Handling StaleElementException

- Definition: Resolves issues when elements are no longer attached to the DOM.
- Example: Refresh the locator or add wait conditions.

```
driver.findElement(By.id("refresh")).click();
```

Selenium

Retrying Failed Tests

- Definition: Retries tests automatically upon failure using TestNG retry logic.
- Example: Implement IRetryAnalyzer in TestNG.

Selenium Best Practices

  - Write reusable methods for actions (e.g., click, sendKeys).
  - Avoid hardcoded waits; use dynamic waits instead.
  - Use Page Object Model (POM) for better maintainability.

Handling Cookies

- Definition: Adds, deletes, or retrieves browser cookies during tests.
- Example:

```
Cookie cookie = new Cookie("key", "value");
driver.manage().addCookie(cookie);
```

Selenium

Screenshot Capture

- Definition: Captures a screenshot during a test failure or at specific steps.
- Example:

```
File screenshot = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(screenshot, new File("screenshot.png"));
```

Exception Handling in Selenium

- Definition: Handles runtime exceptions like NoSuchElementException.
- Example:

```
try {
    driver.findElement(By.id("login")).click();
} catch (NoSuchElementException e) {
    System.out.println("Element not found");
}
```

Logging in Selenium

- Definition: Implements logging frameworks like Log4j for better traceability.
- Example:

```
Logger logger = Logger.getLogger("TestLog");
logger.info("Test Started");
```

Selenium

Allure Reporting

- Definition: Generates interactive and detailed test reports.
- Example: Integrate Allure in Maven by adding its plugin

Assertions in Selenium

- Definition: Validates expected results during a test, using TestNG or JUnit.
- Example (TestNG):

```
Assert.assertEquals(driver.getTitle(), "Expected Title");
```

Browser Profiles

- Definition: Configures browser settings (e.g., disabling pop-ups or adding extensions).
- Example (Firefox):

```
FirefoxOptions options = new FirefoxOptions();
ProfilesIni profile = new ProfilesIni();
options.setProfile(profile.getProfile("default"));
WebDriver driver = new FirefoxDriver(options);
```

Selenium

Scroll Operations

- Definition: Scrolls the webpage to a specific element or position using JavaScript.
- Example:

```
JavascriptExecutor js = (JavascriptExecutor) driver;
js.executeScript("window.scrollBy(0,500)"); // Scroll down by 500px
```

Handling Tooltips

- Definition: Captures and validates tooltips displayed when hovering over elements.
- Example:

```
WebElement element = driver.findElement(By.id("tooltip"));
String tooltipText = element.getAttribute("title");
System.out.println("Tooltip: " + tooltipText);
```

Session Management

- Definition: Manages browser sessions using cookies or WebDriver capabilities.
- Example:

```
String sessionId = ((RemoteWebDriver) driver).getSessionId().toString();
System.out.println("Session ID: " + sessionId);
```

Selenium

Database Testing with Selenium

- Definition: Connects to databases to validate backend operations during tests.
- Example:

```
Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/testdb"
    , "user", "password");
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM users");
```

Email Testing with Selenium

- Definition: Automates email verification by accessing inboxes during testing.
- Example: Use JavaMail API to fetch emails and validate email contents.

API Testing in Parallel with Selenium

- Definition: Integrates REST API testing with tools like RestAssured in Selenium projects.
- Example: Validate API response and compare results with UI behavior.

Selenium

Handling ElementNotInteractableException

- Definition: Occurs when trying to interact with hidden or unclickable elements.
- Example: Use JavaScriptExecutor to click instead:

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

Custom Retry Logic

- Definition: Implements custom logic to retry failed tests beyond built-in frameworks.
- Example (TestNG):

```
public class RetryAnalyzer implements IRetryAnalyzer {
    private int retryCount = 0;
    private static final int maxRetryCount = 3;
    public boolean retry(ITestResult result) {
        if (retryCount < maxRetryCount) {
            retryCount++;
            return true;
        }
        return false;
    }
}
```

Thank You

Share your comments