Selenium:

**What is Selenium?**

Selenium is a free (open source) automated testing suite for web applications across different browsers and platforms. It is quite similar to HP Quick Test Pro (QTP now UFT) only that Selenium focuses on automating web-based applications. Testing done using Selenium tool is usually referred as Selenium Testing.

Selenium is not just a single tool but a suite of software's, each catering to different testing needs of an organization. **It has four components.**

* Selenium Integrated Development Environment (IDE)
* Selenium Remote Control (RC)
* WebDriver
* Selenium Grid



**How to Choose the Right Selenium Tool for Your Need**

| **Tool** | **Why Choose?** |
| --- | --- |
| **Selenium IDE** | * To learn about concepts on automated testing and Selenium, including: * Selenese commands such as type, open, clickAndWait, assert, verify, etc. * Locators such as id, name, xpath, css selector, etc. * Executing customized JavaScript code using runScript * Exporting test cases in various formats. * To create tests with little or no prior knowledge in programming. * To create simple test cases and test suites that you can export later to RC or WebDriver. * To test a web application against Firefox and Chrome only. |
| **Selenium RC** | * To design a test using a more expressive language than Selenese * To run your test against different browsers (except HtmlUnit) on different operating systems. * To deploy your tests across multiple environments using Selenium Grid. * To test your application against a new browser that supports JavaScript. * To test web applications with complex AJAX-based scenarios. |
| **WebDriver** | * To use a certain programming language in designing your test case. * To test applications that are rich in AJAX-based functionalities. * To execute tests on the HtmlUnit browser. * To create customized test results. |
| **Selenium Grid** | * To run your Selenium RC scripts in multiple browsers and operating systems simultaneously. * To run a huge test suite, that needs to complete in the soonest time possible |

**Importing Packages**

To get started, you need to import following two packages:

1. **org.openqa.selenium.\***- contains the WebDriver class needed to instantiate a new browser loaded with a specific driver
2. **org.openqa.selenium.firefox.FirefoxDriver**- contains the FirefoxDriver class needed to instantiate a Firefox-specific driver onto the browser instantiated by the WebDriver class
3. **org.openqa.selenium.chrome.ChromeDriver-** contains the ChromeDriver class needed to instantiate a chrome-specific driver onto the browser instantiated by the WebDriver class.

## Instantiating objects and variables

Normally, this is how a driver object is instantiated.

[First Selenium Webdriver Script: JAVA Code Example](https://www.guru99.com/images/image004(2).png)

A FirefoxDriver class with no parameters means that the default Firefox profile will be launched by our Java program. The default Firefox profile is similar to launching Firefox in safe mode (no extensions are loaded).

For convenience, we saved the Base URL and the expected title as variables.

## Launching a Browser Session

WebDriver's **get()** method is used to launch a new browser session and directs it to the URL that you specify as its parameter.

[First Selenium Webdriver Script: JAVA Code Example](https://www.guru99.com/images/image005(2).png)

**Get Commands**

Get commands fetch various important information about the page/element. Here are some important "get" commands you must be familiar with.

|  |  |
| --- | --- |
| **get()** *Sample usage:* | * It automatically opens a new browser window and fetches the page that you specify inside its parentheses. * It is the counterpart of Selenium IDE's "open" command. * The parameter must be a **String** object. |
| **getTitle()** *Sample usage:* | * Needs no parameters * Fetches the title of the current page * Leading and trailing white spaces are trimmed * Returns a null string if the page has no title |
| **getPageSource()** *Sample usage:* | * Needs no parameters * Returns the **source code of the page** as a String value |
| **getCurrentUrl()** *Sample usage:* | * Needs no parameters * Fetches the string representing the **current URL** that the browser is looking at |
| **getText()** *Sample usage:* | * Fetches the **inner text** of the element that you specify |

**Navigate commands**

These commands allow you to  refresh,go-into and switch back and forth between different web pages.

|  |  |
| --- | --- |
| **navigate().to()** *Sample usage:* | * It automatically **opens a new browser window and fetches the page** that you specify inside its parentheses. * **It does exactly the same thing as the get() method.** |
| **navigate().refresh()** *Sample usage:* | * Needs no parameters. * It **refreshes** the current page. |
| **navigate().back()** *Sample usage:* | * Needs no parameters * Takes you **back by one page** on the browser's history. |
| **navigate().forward()** *Sample usage:* | * Needs no parameters * Takes you **forward by one page** on the browser's history. |

**Closing and Quitting Browser Windows**

|  |  |
| --- | --- |
| **close()** *Sample usage:* | * Needs no parameters * **It closes only the browser window that WebDriver is currently controlling**. |
| **quit()** *Sample usage:* | * Needs no parameters * **It closes all windows that WebDriver has opened.** |

Locators: <https://www.guru99.com/locators-in-selenium-ide.html>

|  |  |  |
| --- | --- | --- |
| **Variation** | **Description** | **Sample** |
| By.**className** | finds elements based on the value of the "class" attribute | findElement(By.className("someClassName")) |
| By.**cssSelector** | finds elements based on the driver's underlying CSS Selector engine | findElement(By.cssSelector("input#email")) |
| By.**id** | locates elements by the value of their "id" attribute | findElement(By.id("someId")) |
| By.**linkText** | finds a link element by the exact text it displays | findElement(By.linkText("REGISTRATION")) |
| By.**name** | locates elements by the value of the "name" attribute | findElement(By.name("someName")) |
| By.**partialLinkText** | locates elements that contain the given link text | findElement(By.partialLinkText("REG")) |
| By.**tagName** | locates elements by their tag name | findElement(By.tagName("div")) |
| By.**xpath** | locates elements via XPath | findElement(By.xpath("//html/body/div/table/tbody/tr/td[2]/table/ tbody/tr[4]/td/table/tbody/tr/td[2]/table/tbody/tr[2]/td[3]/ form/table/tbody/tr[5]")) |

# Find Element and FindElements in Selenium WebDriver

**Find Element:**

Interaction with a web page requires a user to locate the web element. Find Element command is used to uniquely identify a (one) web element within the web page. Whereas, Find Elements command is used to uniquely identify the list of web elements within the web page.

**FindElements:**

Find Elements command takes in By object as the parameter and returns a list of web elements. It returns an empty list if there are no elements found using the given locator strategy and locator value. Below is the syntax of find elements command.

| **Element** | **Command** | **Description** |
| --- | --- | --- |
| **Input Box** | sendKeys() | used to enter values onto text boxes |
| clear() | used to clear text boxes of its current value |
| **Links** | click() | used to click on the link and wait for page load to complete before proceeding to the next command. |
| **Submit Button** | submit() |  |

Sample code:

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.\*;

public class Form {

public static void main(String[] args) {

// declaration and instantiation of objects/variables

System.setProperty("webdriver.chrome.driver","G:\\chromedriver.exe");

WebDriver driver = new ChromeDriver();

String baseUrl = "http://demo.guru99.com/test/login.html";

driver.get(baseUrl);

// Get the WebElement corresponding to the Email Address(TextField)

WebElement email = driver.findElement(By.id("email"));

// Get the WebElement corresponding to the Password Field

WebElement password = driver.findElement(By.name("passwd"));

email.sendKeys("[abcd@gmail.com](mailto:abcd@gmail.com)");

password.sendKeys("abcdefghlkjl");

System.out.println("Text Field Set");

// Deleting values in the text box

email.clear();

password.clear();

System.out.println("Text Field Cleared");

// Find the submit button

WebElement login = driver.findElement(By.id("SubmitLogin"));

// Using click method to submit form

email.sendKeys("[abcd@gmail.com](mailto:abcd@gmail.com)");

password.sendKeys("abcdefghlkjl");

login.click();

System.out.println("Login Done with Click");

//using submit method to submit the form. Submit used on password field

driver.get(baseUrl);

driver.findElement(By.id("email")).sendKeys("[abcd@gmail.com](mailto:abcd@gmail.com)");

driver.findElement(By.name("passwd")).sendKeys("abcdefghlkjl");

driver.findElement(By.id("SubmitLogin")).submit();

System.out.println("Login Done with Submit");

//driver.close();

}

}

**Check Box,** **Radio Button**

| **Element** | **Command** | **Description** |
| --- | --- | --- |
| **Check Box,** **Radio Button** | click() | used to toggle the element on/off |

## Select Option from Drop-Down Box

1. Import the package **org.openqa.selenium.support.ui.Select**
2. Instantiate the drop-down box as a "Select" object in WebDriver

<https://www.guru99.com/select-option-dropdown-selenium-webdriver.html>

# Mouse Click & Keyboard Event: Action Class in Selenium Webdriver

<https://www.guru99.com/keyboard-mouse-events-files-webdriver.html>