Day - 1

Introduction on Selenium

Selenium is an open-source framework for automated browser testing and web application automation, offering a suite of tools to interact with web browsers programmatically.

Different Selenium components

Selenium comprises four main components:

1. Selenium IDE
2. Selenium RC
3. Selenium WebDriver
4. Selenium Grid

Selenium WebDriver: Allows programmatically interacting with web browsers to

automate testing

Selenium IDE (Integrated Development Environment): A browser

extension for recording and playback of browser interactions

Selenium RC (Remote Control) was the predecessor to WebDriver, providing a server-based architecture for automated web testing but is now deprecated in favor of WebDriver.

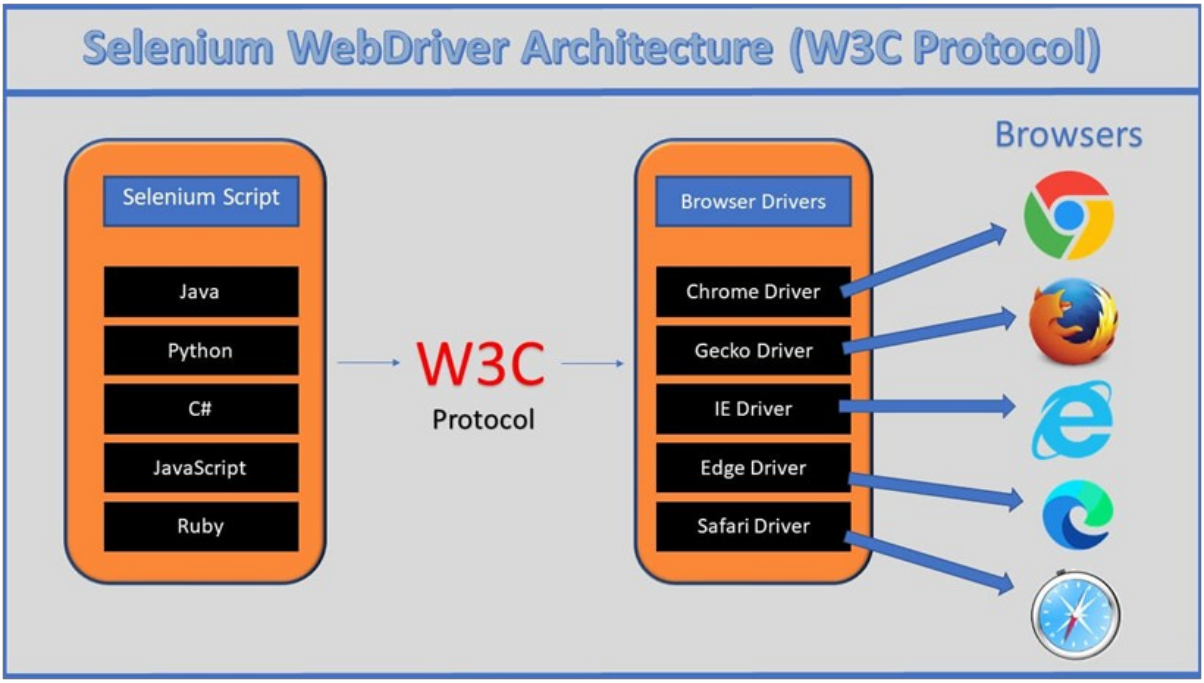
Selenium Grid: Enables parallel test execution across multiple machines and

browsers.

Selenium Architecture

Selenium WebDriver API provides communication facility between languages and browsers.

The following image shows the architectural representation of Selenium WebDriver.



Selenium Language Bindings / Selenium Client Libraries

Selenium 4 supports various programming languages, and developers use Selenium client libraries to write automation scripts. Common languages include Java, Python, C#, Ruby, and JavaScript

W3C WebDriver Protocol

The WebDriver protocol is the actual communication protocol used between the client (automation script or test code) and browser driver. It defines how commands are sent to the browser, how the browser processes those commands, and how the results are communicated back to the client

Browser Drivers

Each browser (e.g., ChromeDriver, GeckoDriver, etc.) has its own driver that implements the WebDriver protocol. These drivers act as intermediaries between the Selenium scripts and the browsers, translating commands from the script into actions in the browser

Hierarchy of Selenium Webdriver.

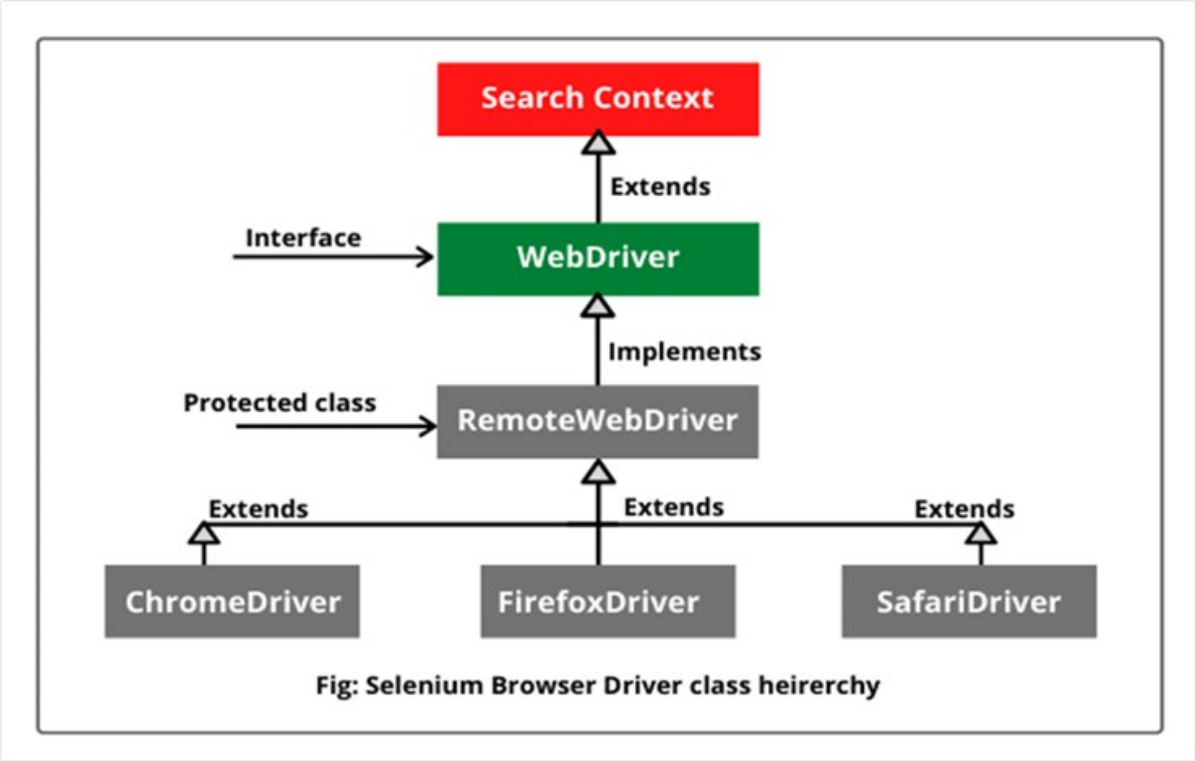


Fig: Selenium Browser Driver class heirerchy

SearchContext - is an Interface with 2 Abstract methods - FindElement() and FindElements().

WebDriver - is an interface which has 11+2 abstract methods.

RemoteWebDriver - is a protected class which is an implementation class for webdriver. It has 13 concrete methods.

Remote webdriver implements Webdriver.

ChromeDriver - A WebDriver implementation that controls a Chrome browser running on the local machine.

It requires a chromedriver executable to be available in PATH**.**

ChromeDriver extends RemoteWebDriver

Selenium Installation

Installation of Selenium web driver, making sure all Environmental variables are set and installing selenium library in IDE.

How to install Selenium -

1) Downloaded Selenium Webdriver from Official Selenium Website.
2) Extracted Selenium files
3) In Eclipse, Properties of Project → Java Build path → Libraries —> Add External JARS→ Selected and opened all the Selenium files that we extracted.--> Apply and close.
4) Download Chromedriver (Make sure to download the latest version of chromedriver by checking against your chrome version / the chrome version used in your project)
5) We include the chromedriver file path in your script.

First Selenium Script :

```
public class FirstScript {

        public static void main(String[] args) {

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

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

  WebDriver driver = new ChromeDriver();

   driver.get("https://www.google.com/");

      }

}
```

System.setproperty - It is used to enter the details of the driver(chromedriver,firefoxdriver,edgedriver) in a key value pair of Setproperty(key, value) key is the driver detail, value is the location of the driver.

Web driver is an **Interface**, driver is the reference variable, new is the keyword, ChromeDriver is the Constructor, together a new **ChromeDriver**() object is created and stored in the reference variable **driver**.

Here, All the driver methods are just declared in **Webdriver** Interface, and all the driver methods(get,close,quit,findelement etc) are implemented in Implementation class called **ChromeDriver**();

Locators

In Automation , before performing any action such as click, clear, pass the data (send keys) we need to find the address of the element.

We do that by inspecting the element

To find the element, we use Locators

Locators are classified into 8 types.

1) ClassName.
2) CSS selector.
3) ID.
4) LinkText.
5) Name
6) PartialLinktext.
7) TagName
8) X path.

All of the Locators are static methods of By class.

By class is an abstract class.

Official info about Locators in **Selenium website** :

https://www.selenium.dev/documentation/webdriver/elements/locators/

What is a CSS Selector?

CSS (Cascading Style Sheets) Selectors in Selenium are used to identify and locate web elements based on their id, class, name, attributes and other attributes.

CSS is a preferred locator strategy as it is simpler to write and faster as compared to XPath.

By.cssSelector(String cssSelector) method is used to locate the elements in Selenium WebDriver.

This method accepts a CSS Selector String as an argument which defines the selection method for the web elements.

Types of CSS Selectors in Selenium (with Examples)
There are five types of CSS Selectors in Selenium tests:

1. ID
2. Class
3. Attribute

-
1. ID

In CSS, we can use "**#**" notation to select the "**id**" attribute of an element.

Syntax:

<tagname>#<id_value>

#<id_value>

2. Class

In CSS, we can use "." notation to select the "class" attribute of an element.

Syntax:

<tagname>.<class_value>

.<class value>

3. **Attribute**

Apart from "**id**" and "**class**", other attributes can also be used to locate web elements using CSS selector.

Syntax:

<tagname>[attribute_name='<attribute_value>']

Programs :

```
package Day1;

import org.openqa.selenium.SearchContext;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.remote.RemoteWebDriver;

public class FIrstScript {

	public static void main(String[] args) {

		//Step 1 - set property of chromedriver - path

		//System.setProperty("Webdriver.chrome.driver","C:\\Users\\Dell\\Downloads\\
chromedriver-122\\chromedriver.exe");


		// Step 2 -	// It will open a new chrome browser

		WebDriver driver = new ChromeDriver();

		//WebDriver driver = new ChromeDriver(); // Then our code will only be limited
to chrome browser


		//RemoteWebDriver driver = new ChromeDriver(); // Selenium grid is used for
Remote execution

		driver.get("https://www.google.com");
	}

}
```

```
package Day1;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
```

```
public class herokuLogin {

	public static void main(String[] args) {


		WebDriver driver = new ChromeDriver(); // open the browser

		driver.get("https://the-internet.herokuapp.com/login"); // passing the URL

		driver.findElement(By.name("username")).sendKeys("Omkar");
		// sendkeys is a method to send some data inside textbox

		driver.findElement(By.id("password")).sendKeys("omkar@2002");

		//driver.findElement(By.tagName("i")).click();

		driver.findElement(By.cssSelector("i.fa-sign-in")).click();


		//driver.findElement(By.linkText("Elemental Selenium")).click();

		//driver.findElement(By.partialLinkText("Elemental"));




	}

}
```

```
package Day1;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class fblogin {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		WebDriver driver = new ChromeDriver();

		driver.get("https://www.facebook.com/login/");


		driver.findElement(By.cssSelector("#email")).sendKeys("sdasdasf");
```

```
                driver.findElement(By.cssSelector(""));


        }

}
```