Contents

Selenium

Selenium is one of the most widely used open source Web UI automation testing tools. It supports automation testing of websites across different browsers such as Edge, Chrome, Safari and Firefox etc. It supports various platforms and programming languages. User can automate functional tests and easily integrates them with Maven, Jenkins and other build automation and continuous integration tools.

Selenium Suite Components

Selenium IDE

It is an add-on to Firefox browser that provides record and playback capability. It has limited features and test scripts generated are not robust and portable.

Selenium RC

Selenium Remote Control (RC) is officially deprecated by Selenium. It uses a dedicated server for running automation scripts.

Selenium WebDriver

It provides different drivers for different browsers and support different programming languages.

Selenium Grid

It helps in running selenium test scripts in parallel mode across multiple remote machines.

Why Selenium

1. Open Source
2. Scripting can be done in many programming languages like C#, Java, Python, Perl and Ruby etc.
3. Supports all popular browsers like Edge, Firefox, Chrome, IE, Opera and Safari etc.
4. Selenium grid helps in parallel and distributed test execution.

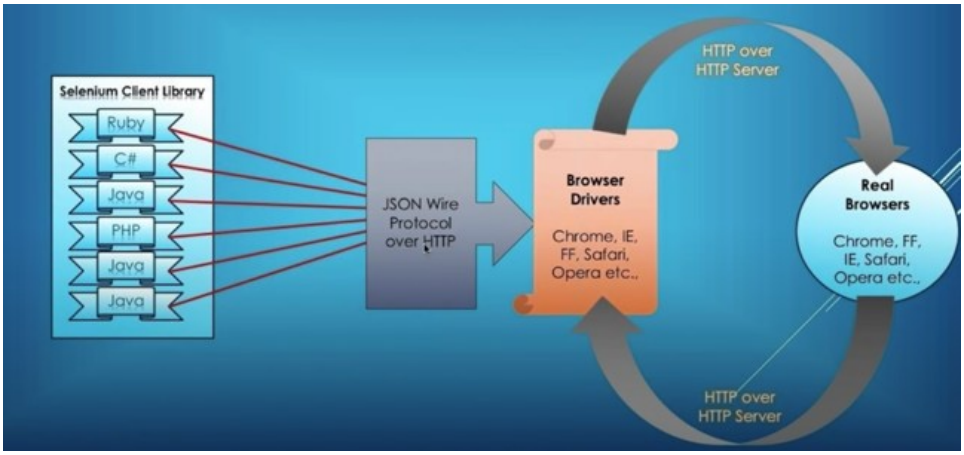
Limitations of Selenium

1. Selenium does not support desktop application test automation. User need to use Robot Class or AutoIT for performing any windows based actions as part of Web Automation testing.
2. REST or SOAP web services cannot be automated
3. For reporting / logging, reading-writing external files rely on external libraries (TestNG, Extent Report, Allure Reports and POI Jars for excel file reading etc.)

How Selenium WebDriver works

Selenium WebDriver (Selenium 2.0) automates the browsers actions by calling their native events (Click, MouseOver, KeyPress etc.) directly unlike Selenium RC which injects Javascript in browsers for test automation. WebDriver is much faster than Selenium RC and also handles scenarios like alerts, pop-ups, Keyboard and Mouse actions easily. Some of the widely used drivers in Selenium are ChromeDriver, FirefoxDriver, InternetExplorerDriver, SafariDriver and HtmlUnitDriver (headless driver) etc.

Selenium Architecture



About JSON Wire Protocol

https://www.selenium.dev/documentation/legacy/json_wire_protocol/

Selenium First Program

Required software for Selenium Program

1. Install Java
2. Download Eclipse (ensure Maven plugin is available)

Additional details:

Selenium JAR from https://www.selenium.dev/downloads/

Browser Driver from https://chromedriver.chromium.org/

Steps to create Selenium Program

1. Open Eclipse and create a simple Maven Project
2. Update POM.xml with Selenium-Java and WebDriverManager dependency

OnlinePortal/pom.xml

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>walmart</groupId>
    <artifactId>OnlinePortal</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <dependencies>
        <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <version>4.3.0</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->
        <dependency>
            <groupId>io.github.bonigarcia</groupId>
            <artifactId>webdrivermanager</artifactId>
            <version>5.2.1</version>
        </dependency>
    </dependencies>
</project>
```

First Program

SeleniumDemo.java

```java
package day23;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.edge.EdgeDriver;

import io.github.bonigarcia.wdm.WebDriverManager;

public class SeleniumDemo {
    public static void main(String[] args) {
        WebDriverManager.chromedriver().setup();
        WebDriver driver = new ChromeDriver();
        driver.get("https://www.google.co.in");
        driver.close();
        driver.quit();

        WebDriverManager.edgedriver().setup();
        WebDriver driver2 = new EdgeDriver();
        driver2.get("https://www.intel.com");
        driver2.close();
        driver2.quit();
    }
}
```

Notes:

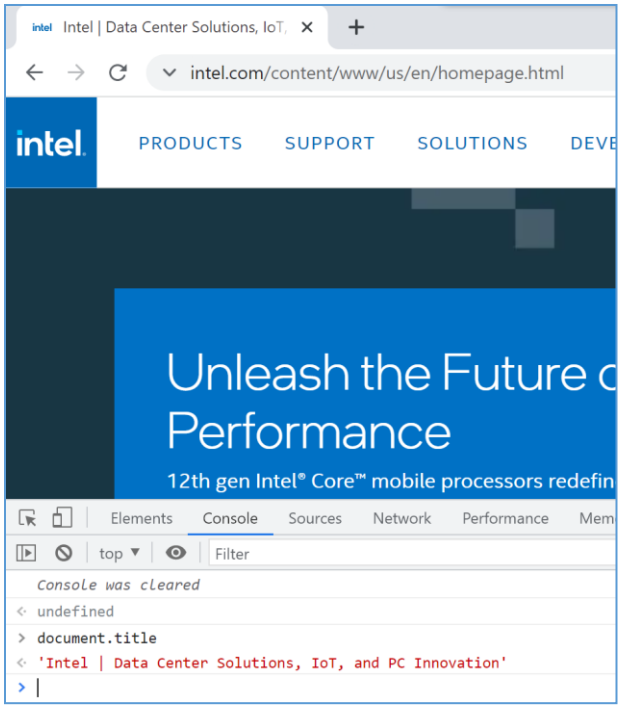
1. WebDriver is an interface. It has empty methods but not implemented. These methods can be implements anyone until method type and signatures are not violated.
2. ChromeDriver is a class that has been written specially for Chrome Browser. It has many methods and implemented and can be instantiated. ChromeDriver can perform all actions on the Chrome browser as defined in the interface WebDriver.

Browser Developer Tools

Use F12 function key to get a Browser Developer tools.

Browser Console

Click on Console Tab to give any browser (Document) related commands like below:



Object Identification

Selenium By Class

Selenium By Class used to locate elements within a document using className, cssSelector, equals, hashCode, id, linkText, name, partialLinkTest, tagName and **xpath**.

Note: Ensure your xpath is able to identify your application WebElement uniquely.

XPath

XPath(XML Path Language) is based on a tree representation of the XML document and used for selecting nodes the document. It also provides ability to navigate around the tree.

1. Absolute XPath – A single "/" at the beginning instructs XPath engine start looking for an element from root node.
   /html/body/div[2]/div/div/footer/section[3]/div/ul/li[3]/a
2. Relative XPath – A double "//" instructs XPath engine look for an element anywhere in the XML document for given criteria.
   Syntax: //TagName[@AttributeName='Attribute Value']
   Examples:
   //*[@id='lst-ib']
   //button[@label='Continue']
3. Partial XPath – If locators properties changing frequently then try with "Contains" keyword.
   //div[contains(@class, 'footer-nav')]
   //h1[text()='Mobile Deposit Confirmation']

Partial XPath Keyword

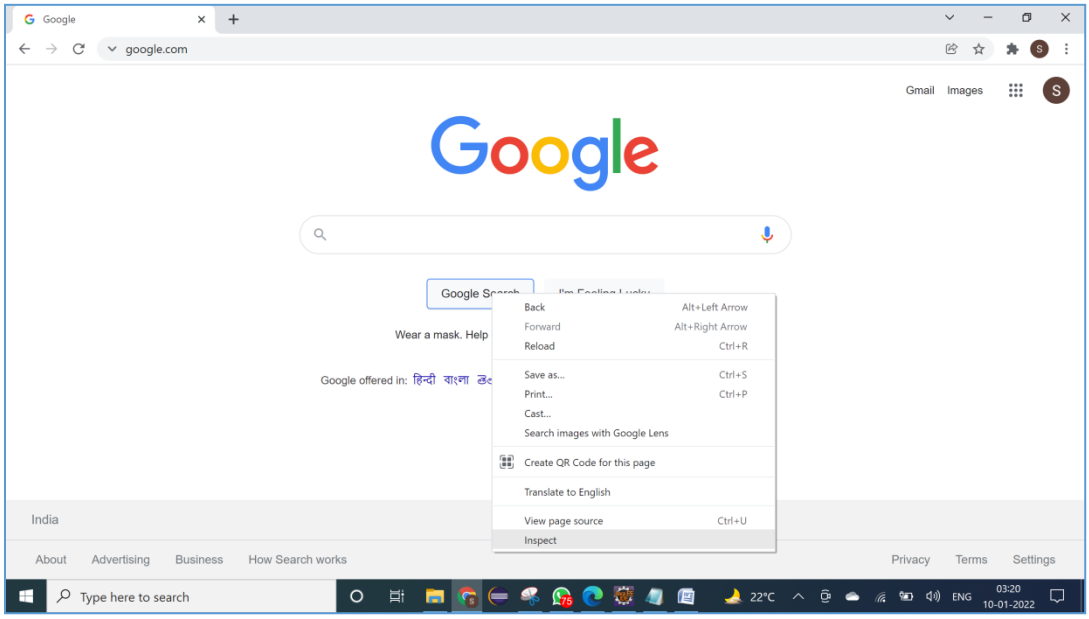
1. Contains
2. Text

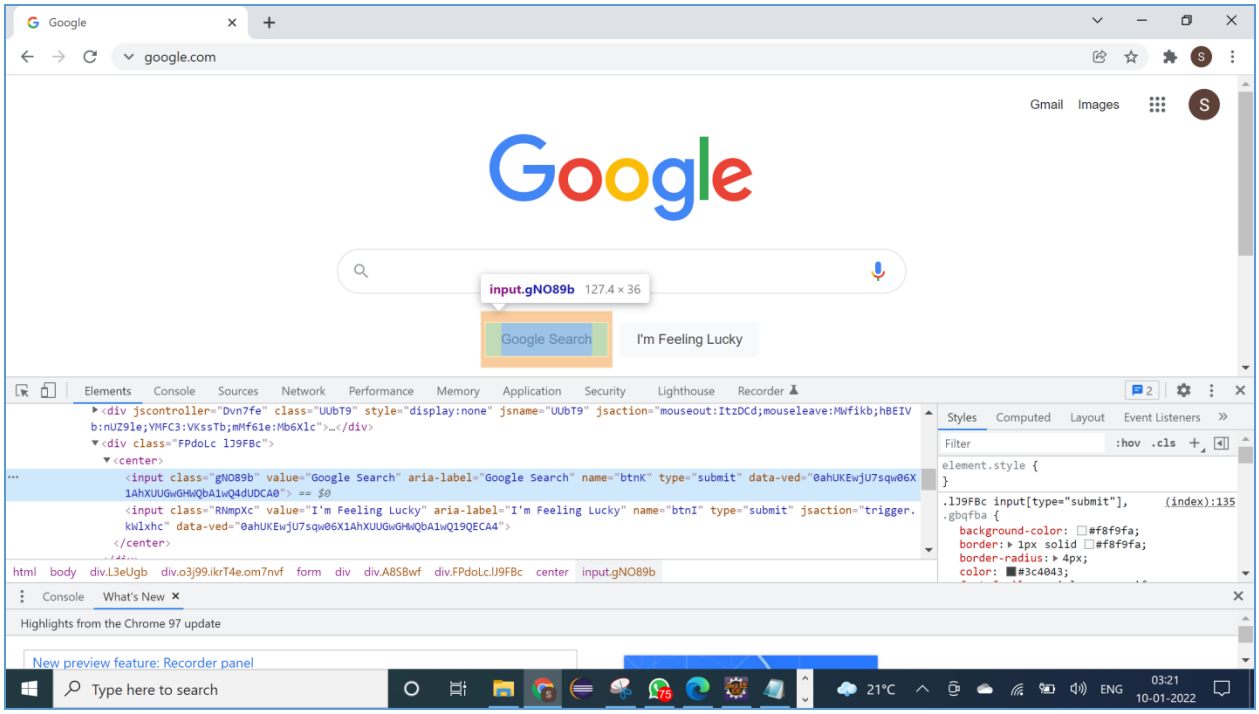
3. Start-with

DOM – Document Object Model

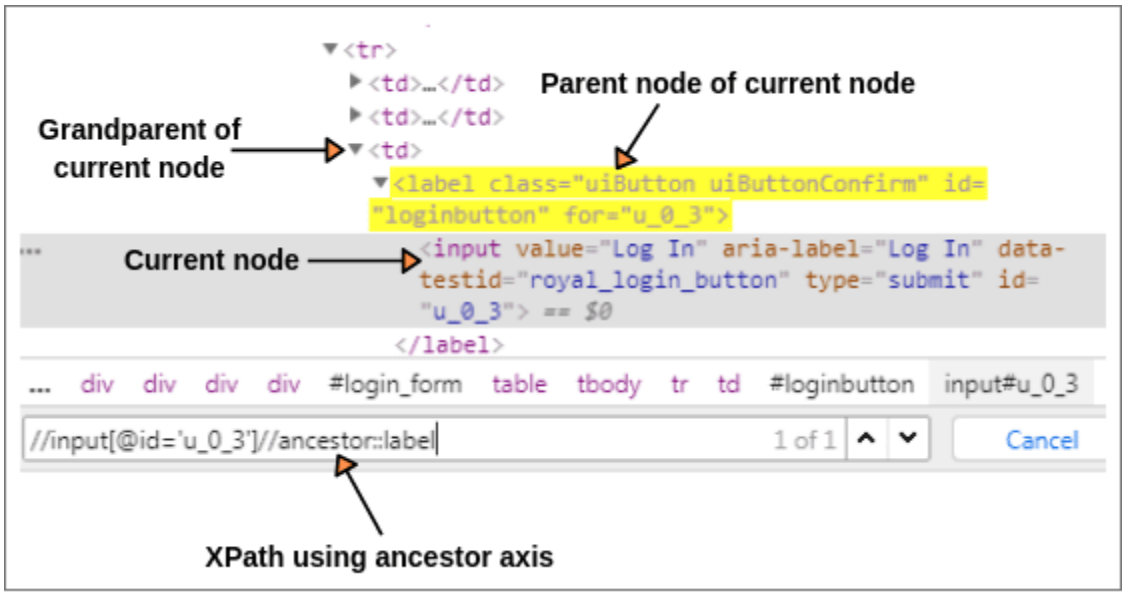
To write xpath of any WebElement in a Chrome browser follow below steps:

1. Open browser and launch your application URL
2. Mouse Hover on WebElement you want to get xpath
3. Mouse Right click on the WebElement and click on Inspect from shortcut menu
4. You will get DOM of the webpage





XPath Axes



WebElement Class

Anything that is present on the web page is a WebElement such as text box, button, etc. WebElement represents an HTML element. Selenium WebDriver encapsulates a simple form element as an object of the WebElement. It basically represents a DOM element and all the HTML documents are made up by these HTML elements. Each HTML element consists of a start tag and an end tag. The content lies between the tags.

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

In above example, searchBox is a WebElement and user can perform any below actions:

1. .sendKeys
2. .isDisplayed
3. .isSelected
4. .submit
5. .isEnabled
6. .clear
7. .click

Refer this link for more WebElement commands / actions.

Refer this link to understand various WebElement Exceptions.

Actions Class

Selenium Actions class is used to perform Mouse hover events, keyboard events. User can use this class for advanced actions such as drag and drop, multi selection etc.

ActionsClassDemo.java

```
package selenium.basics;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.interactions.Actions;
public class ActionsClassDemo {
    public static void main(String[] args) throws InterruptedException {
        // Set Chrome Drive path
        System.setProperty("webdriver.chrome.driver", ".//Resources//chromedriver.exe");
        // Create driver objet
        WebDriver driver = new ChromeDriver();
        // Launch application
        driver.get("https://www.google.co.in");
        // Static wait
        Thread.sleep(3000);
        // Create WebElement for Search Text box
        WebElement txtBox = driver.findElement(By.name("q"));
        // Type Automation Testing in the search Text box
        txtBox.sendKeys("Automation Testing");
        // Create WebElement for Google search button
        WebElement btnSearch = driver.findElement(By.name("btnK"));
        // Static Wait
        Thread.sleep(2000);
        // Create Actions class object
        Actions actObj = new Actions(driver);
        // Perform click action on Google Search button using Actions class
        actObj.moveToElement(btnSearch).click().perform();
        // Static wait
        Thread.sleep(2000);
        // Close Browser tab
        driver.close();
        // Quit Browser
        driver.quit();
    }
}
```

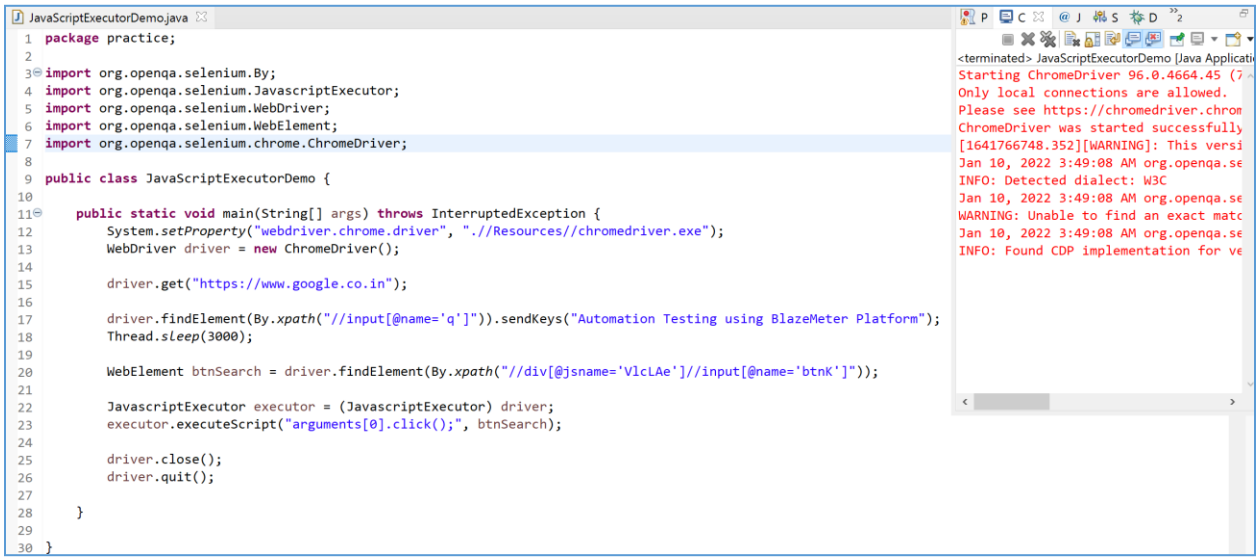
Another Example:

```
public class ActionsDemo {

	public static void main(String[] args) throws InterruptedException {
		System.setProperty("webdriver.chrome.driver", ".//Resources//chromedriver.exe");
		WebDriver driver  = new ChromeDriver();
		driver.manage().window().maximize();
		Actions act = new Actions(driver);
		driver.get("https://www.online.citibank.co.in/");
		Thread.sleep(2000);
		WebElement lnkCreditCards = driver.findElement(By.xpath("//a[@title='Credit Cards' and @id='topMnucreditcards']"));
		act.moveToElement(lnkCreditCards).perform();
		Thread.sleep(10000);
		driver.close();
		driver.quit();
	}
}
```

JavascriptExecutor

Sometimes Selenium WebDriver gives ElementNotInteractableException then user might need JavascriptExecutor. Please look at below URL for more information.

https://www.browserstack.com/guide/javascriptexecutor-in-selenium



Scroll Up and Scroll Down using JavascriptExecutor

```
//scroll down
js.executeScript("window.scrollBy(0,250)", "");

//Scroll up
js.executeScript("window.scrollBy(0,-350)", "");
```

Move to Element using JavascriptExecutor

```
JavascriptExecutor js = (JavascriptExecutor) driver;
WebElement lnkHelp = driver.findElement(By.xpath("//a[text()='Help']"));
js.executeScript("arguments[0].scrollIntoView();", lnkHelp);
```

Working with Edit / Text box

Use .sendKeys method to type or enter any text in an Edit or Text box of your webpage.

Approach #1:

```
driver.findElement(By.xpath("//input[@name='q']")).sendKeys("Automation Testing using BlazeMeter Platform");
```

Approach #2:

```
WebElement searchBox = driver.findElement(By.xpath("//input[@name='q']"));
searchBox.sendKeys("Automation Testing using BlazeMeter Platform");
```

Actions Class

Selenium Actions class is used to perform Mouse hover events, keyboard events. User can use this class for advanced actions such as drag and drop, multi selection etc.

ActionsClassDemo.java

```
1  package selenium.basics;
2  import org.openqa.selenium.By;
3  import org.openqa.selenium.WebDriver;
4  import org.openqa.selenium.WebElement;
5  import org.openqa.selenium.chrome.ChromeDriver;
6  import org.openqa.selenium.interactions.Actions;
7  public class ActionsClassDemo {
8      public static void main(String[] args) throws InterruptedException {
9          // Set Chrome Drive path
10         System.setProperty("webdriver.chrome.driver", ".//Resources//chromedriver.exe");
11         // Create driver objet
12         WebDriver driver = new ChromeDriver();
13         // Launch application
14         driver.get("https://www.google.co.in");
15         // Static wait
16         Thread.sleep(3000);
17         // Create WebElement for Search Text box
18         WebElement txtBox = driver.findElement(By.name("q"));
19         // Type Automation Testing in the search Text box
20         txtBox.sendKeys("Automation Testing");
21         // Create WebElement for Google search button
22         WebElement btnSearch = driver.findElement(By.name("btnK"));
23         // Static Wait
24         Thread.sleep(2000);
25         // Create Actions class object
26         Actions actObj = new Actions(driver);
27         // Perform click action on Google Search button using Actions class
28         actObj.moveToElement(btnSearch).click().perform();
29         // Static wait
30         Thread.sleep(2000);
31         // Close Browser tab
32         driver.close();
33         // Quit Browser
34         driver.quit();
35     }
36 }
```

Another Example:

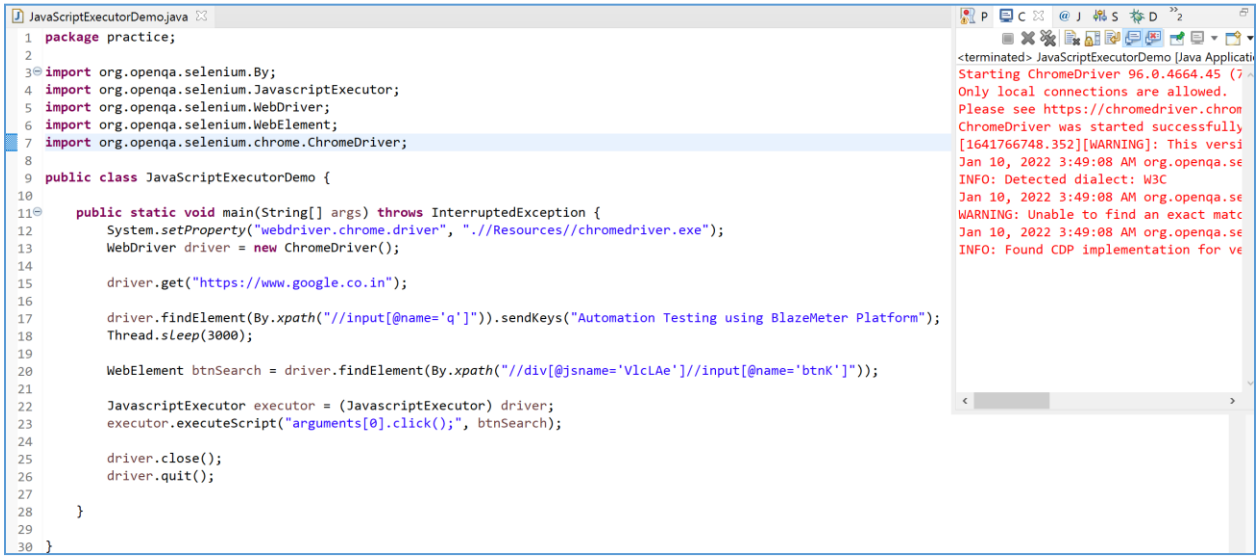
```
public class ActionsDemo {

        public static void main(String[] args) throws InterruptedException {
                System.setProperty("webdriver.chrome.driver", ".//Resources//chromedriver.exe");
                WebDriver driver  = new ChromeDriver();
                driver.manage().window().maximize();
                Actions act = new Actions(driver);
```

```
                driver.get("https://www.online.citibank.co.in/");
                Thread.sleep(2000);
                WebElement lnkCreditCards = driver.findElement(By.xpath("//a[@title='Credit Cards' and @id='topMnucreditcards']"));
                act.moveToElement(lnkCreditCards).perform();
                Thread.sleep(10000);
                driver.close();
                driver.quit();
        }
}
```

JavascriptExecutor

Sometimes Selenium WebDriver gives ElementNotInteractableException then user might need JavascriptExecutor. Please look at below URL for more information.

https://www.browserstack.com/guide/javascriptexecutor-in-selenium



Scroll Up and Scroll Down using JavascriptExecutor

```
//scroll down
js.executeScript("window.scrollBy(0,250)", "");

//Scroll up
js.executeScript("window.scrollBy(0,-350)", "");
```

Move to Element using JavascriptExecutor

```
JavascriptExecutor js = (JavascriptExecutor) driver;
WebElement lnkHelp = driver.findElement(By.xpath("//a[text()='Help']"));
js.executeScript("arguments[0].scrollIntoView();", lnkHelp);
```

Working with Edit / Text box

Use .sendKeys method to type or enter any text in an Edit or Text box of your webpage.

Approach #1:

```
driver.findElement(By.xpath("//input[@name='q']")).sendKeys("Automation Testing using BlazeMeter Platform");
```

Approach #2:

```
WebElement searchBox = driver.findElement(By.xpath("//input[@name='q']"));
searchBox.sendKeys("Automation Testing using BlazeMeter Platform");
```

Working with check box

Use .click() method to check or uncheck a check box.

Use .isSelected() method to get the check box status that is selected or not.

CheckBoxDemo.java

```
3  import org.openqa.selenium.By;
8  public class CheckBoxDemo {
9      public static void main(String[] args) throws InterruptedException {
10         System.setProperty("webdriver.chrome.driver", ".//Resources//chromedriver.exe");
11         WebDriver driver = new ChromeDriver();
12         driver.get("https://www.walmart.com/account/login?vid=oaoh&tid=0&returnUrl=%2F");
13         WebElement chk1 = driver.findElement(By.xpath("//input[@id='remember-me']"));
14         Thread.sleep(2000);
15         ((JavascriptExecutor)driver).executeScript("arguments[0].click();", chk1);
16         Thread.sleep(1000);
17         System.out.println(chk1.isSelected());
18         ((JavascriptExecutor)driver).executeScript("arguments[0].click();", chk1);
19         System.out.println(chk1.isSelected());
20         Thread.sleep(2000);
21         driver.close();
22         driver.quit();
23     }
24 }
```

Problems | Console | Javadoc | Declaration | Search | Servers | Debug

```
<terminated> CheckBoxDemo [Java Application] C:\Program Files\Java\jdk1.8.0_271\bin\javaw.exe  (10-Jan-2022, 4:28:33 am – 4:28:46 am)
ChromeDriver was started successfully.
[1641769116.743][WARNING]: This version of ChromeDriver has not been tested with Chrome version 97.
Jan 10, 2022 4:28:36 AM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Jan 10, 2022 4:28:37 AM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 97, so returning the closest version found: 96
Jan 10, 2022 4:28:37 AM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
INFO: Found CDP implementation for version 97 of 96
false
true
```

Working with option button

Use .click() method to select any option button.

Working with Dropdown list box

```
WebElement dropDown = driver.findElement(By.id("Best Match"));
Select select1 = new Select(dropDown);
select1.selectByIndex(2);
select1.selectByValue("Price Low to High");
System.out.println(select1.getAllSelectedOptions());
System.out.println(select1.getOptions());
System.out.println(select1.getFirstSelectedOption());
```

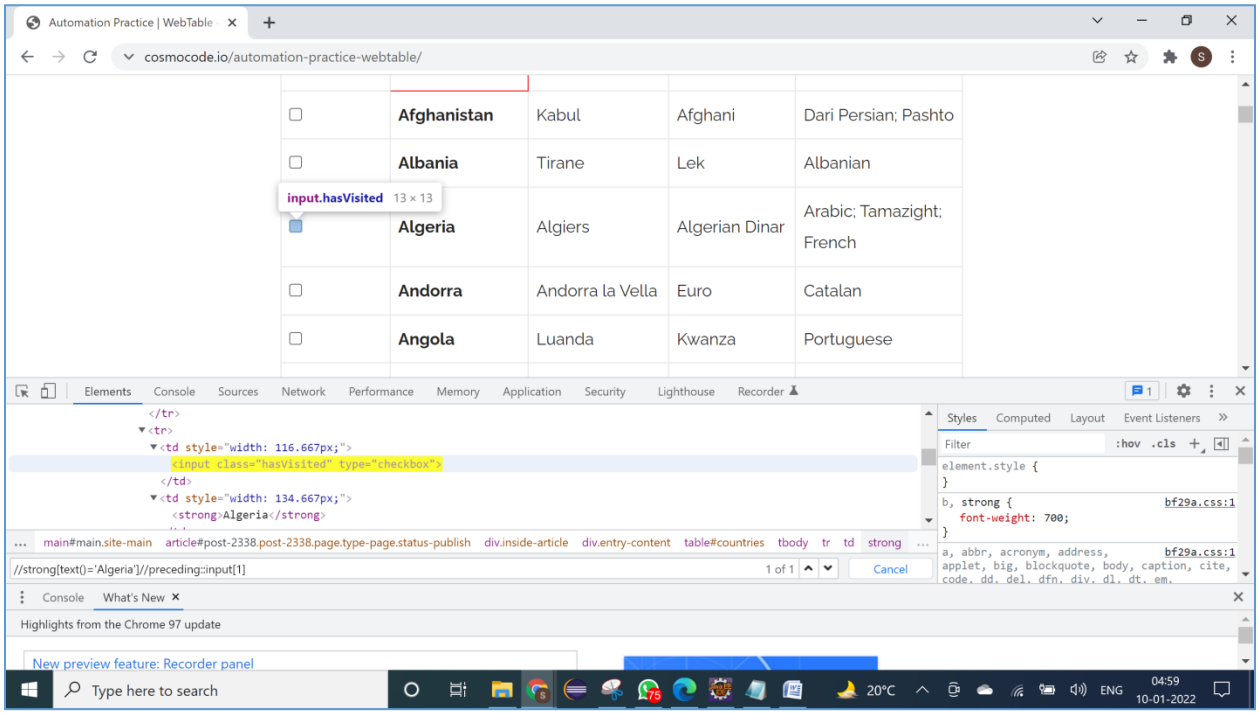
Refer this link for more information.

Working with Web Table

There are multiple ways to interact with a Web Table in a Webpage. User might need to use XPath Axes when there is no unique name. Here is an example:

https://cosmocode.io/automation-practice-webtable/

//strong[text()='Algeria']//preceding::input[1]



Working with Frames

iFrame (Inline Frame) is a web page embedded in a web page. Selenium WebDriver could not find any WebElements inside iFrame directly. User need to switch to iFrame to proceed further testing.

FramesDemo.java

```
9
10  public class FramesDemo {
11
12      public static void main(String[] args) throws InterruptedException {
13          System.setProperty("webdriver.chrome.driver", ".//Resources//chromedriver.exe");
14          WebDriver driver  = new ChromeDriver();
15          driver.manage().window().maximize();
16          driver.get("https://demoqa.com/frames");
17          Thread.sleep(2000);
18
19          driver.switchTo().frame("frame1");
20          List<WebElement> strHeader = driver.findElements(By.xpath("//h1[@id='sampleHeading']"));
21          System.out.println("Total H1 Headers found in frame1: " + strHeader.size());
22          Thread.sleep(2000);
23          driver.close();
24          driver.quit();
25      }
26
```

Problems | Console | Javadoc | Declaration | Search | Servers | Debug

<terminated> FramesDemo [Java Application] C:\Program Files\Java\jdk1.8.0_271\bin\javaw.exe (12-Jan-2022, 6:43:23 pm – 6:43:41 pm)

```
ChromeDriver was started successfully.
[1641993207.547][WARNING]: This version of ChromeDriver has not been tested with Chrome version 97.
Jan 12, 2022 6:43:27 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Jan 12, 2022 6:43:27 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 97, so returning the closest version found: 96
Jan 12, 2022 6:43:27 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
INFO: Found CDP implementation for version 97 of 96
Total H1 Headers found in frame1: 1
```

Capture Screenshot

ScreenshotDemo.java

```java
package practice;

import java.io.File;
import java.io.IOException;

import org.apache.commons.io.FileUtils;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class ScreenshotDemo {

    public static void main(String[] args) throws IOException {
        System.setProperty("webdriver.chrome.driver", ".//Resources//chromedriver.exe");
        String fileName = ".//Resources//screenshot1.png";
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://www.citi.com");
        // Convert web driver object to TakeScreenshot
        TakesScreenshot scrShot = ((TakesScreenshot) driver);
        // Call getScreenshotAs method to create image file
        File SrcFile = scrShot.getScreenshotAs(OutputType.FILE);
        // Move image file to new destination
        File DestFile = new File(fileName);
        // Copy file at destination
        FileUtils.copyFile(SrcFile, DestFile);

        driver.close();
        driver.quit();
    }
```

Navigation Commands

```
driver.navigate().to("www.google.com");
driver.navigate().forward();
driver.navigate().back();
driver.navigate().refresh();
```

Alerts

Frequently used alerts methods in test automation.

```
Alert alert = driver.switchTo().alert();
alert.dismiss();
alert.getText();
alert.accept();
alert.sendKeys("Hello World.");
```

Selenium Wait

Selenium WebDriver provides three commands to implement waits in tests.

1. Implicit Wait

```
driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);  //Sel <= 4
```

```
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10)); //Sel >4.x
```

2. Explicit Wait

```
WebDriverWait explictWait = new WebDriverWait(driver, 20);
```

```
//Selenium 4 syntax
WebDriverWait wait = new WebDriverWait(driver,Duration.ofSeconds(10));
wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector(".classlocator")));
```

3. Fluent Wait

Before Selenium 4 -

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

After Selenium 4 -

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

https://devqa.io/webdriver-explicit-implicit-fluent-wait/

Browser Driver Options

Before Selenium 4 there was DesiredCapabilities class exists and we used these capabilities to define test environment such as OS, Browser name, version etc. In Selenium 4.x replaced with options object.

- Chrome – ChromeOptions
- Microsoft Edge – EdgeOptions
- Firefox – FirefoxOptions
- Safari – SafariOptions
- Internet Explorer (IE) – InternetExplorerOptions

```java
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

public class ChromeOptionsDemo {

    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver", ".//Resources//chromedriver.exe");

        ChromeOptions options = new ChromeOptions();
        //Open Browser in maximized mode
        options.addArguments("start-maximized");
        //Block popups
        options.setExperimentalOption("excludeSwitches",Arrays.asList("disable-popup-blocking"));

        //Set default download directory
        Map<String, Object> prefs = new HashMap<String, Object>();
        prefs.put("download.default_directory", "/directory/path");
        options.setExperimentalOption("prefs", prefs);

        ChromeDriver driver = new ChromeDriver(options);

        driver.get("https://www.google.co.in");
        Thread.sleep(2000);
        driver.close();
        driver.quit();
    }
}
```

Refer this link for more browser options.

Page Object Model

https://www.toolsqa.com/selenium-webdriver/page-object-model/

Page Factory Design Pattern

https://www.browserstack.com/guide/page-object-model-in-selenium

@FindBy to be used while initializing WebElement.

User need to use PageFactory.initElement(driver, this.class) to be used in the page class constructor.

Maven

Build automation tool, download dependencies automatically. There are many other build automation tools available. Example: Gradle, Ant etc.

Maven Lifecycle Phases

https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#a-build-lifecycle-is-made-up-of-phases

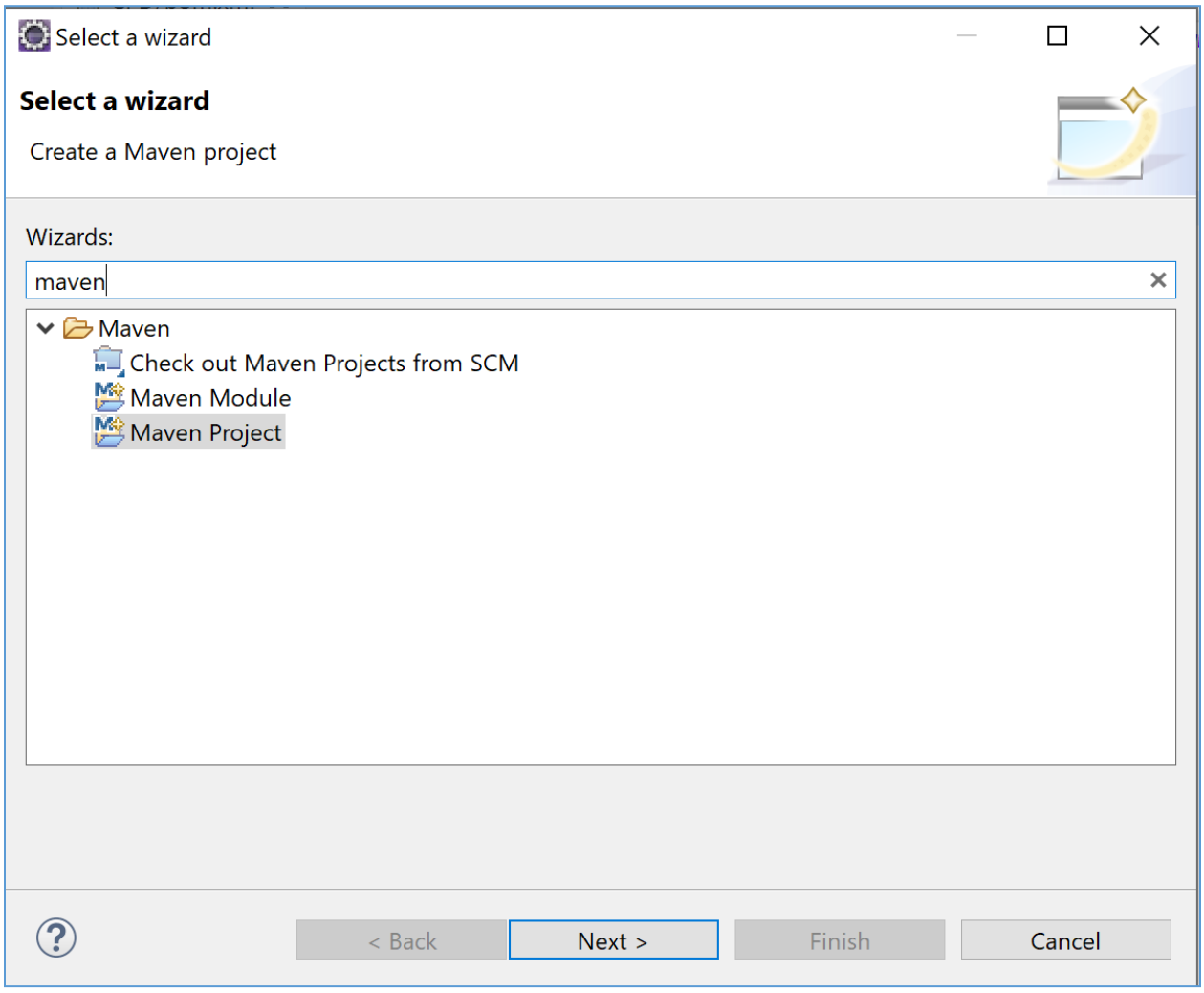
Plug-ins vs Dependencies

Plugin is a Jar file which executes the task, and dependency is a Jar which provides the class files to execute the task.

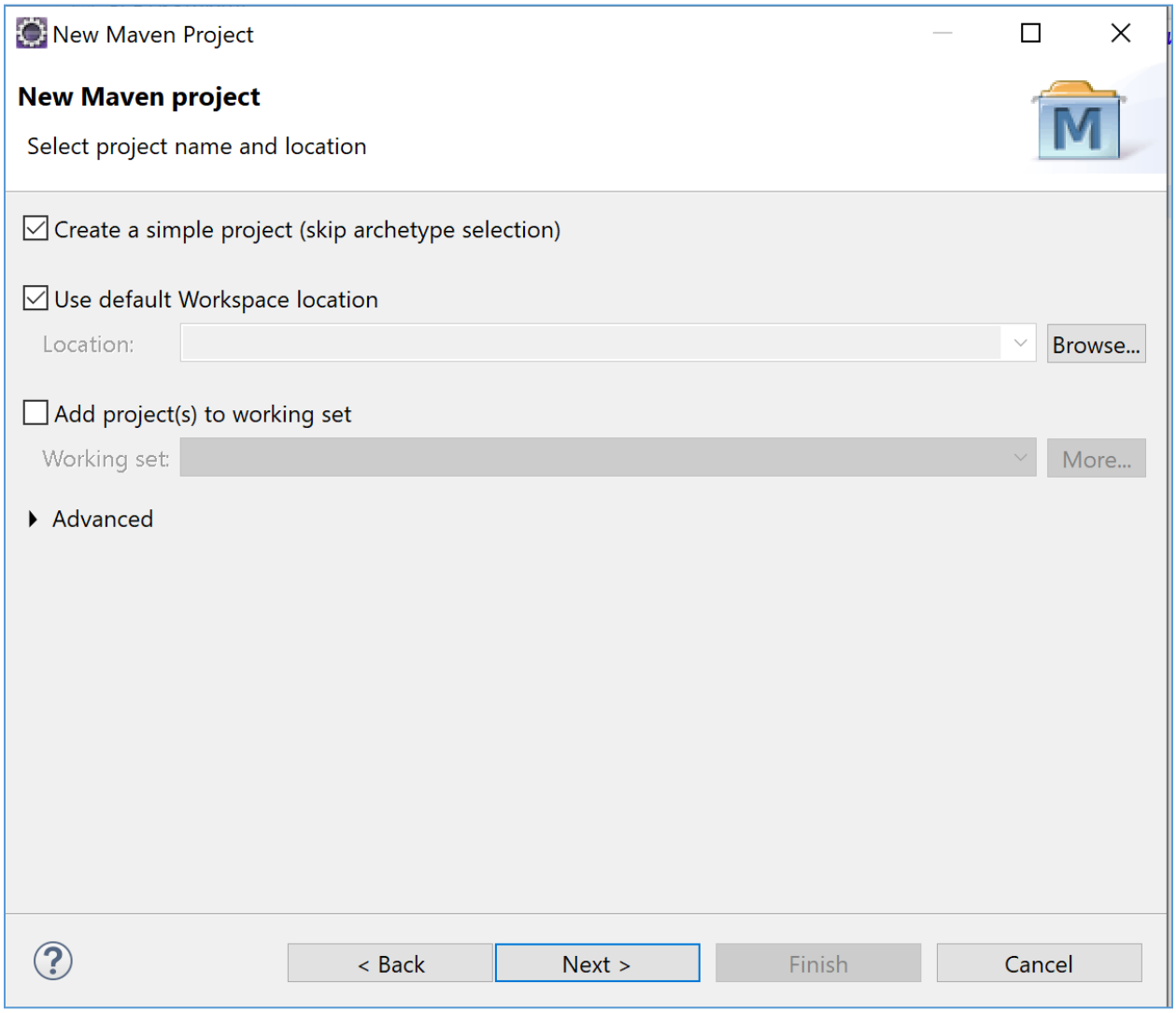
Important points in Maven

1. Maven downloads all dependencies JARs in user profile .m2 folder. Example: C:\users\stata\.m2
2. User must resolve all POM issues before build the Maven project.
3. Maven POM means Project Object Model
4. Maven commands can be executed from CLI (Command Line Interface) too.

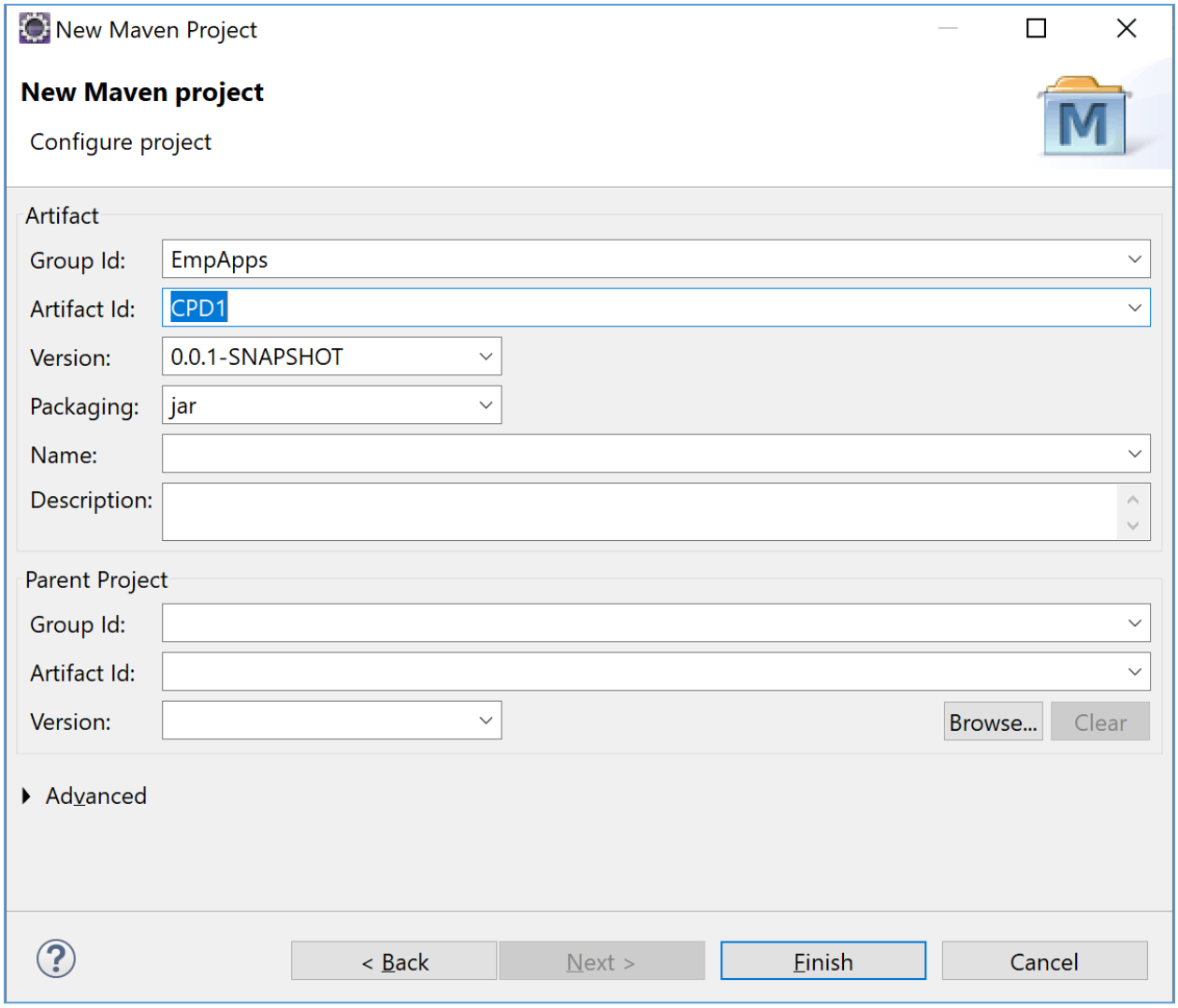
Step 1:



Step 2:


New Maven Project
New Maven project
Select project name and location
Create a simple project (skip archetype selection)
Use default Workspace location
Location:
Browse...
Add project(s) to working set
Working set:
More...
Advanced
< Back
Next >
Finish
Cancel

Step 3:



WebDriverManager

WebDriverManager is an open-source Java library that carries out the management (i.e., download, setup, and maintenance) of the drivers required by Selenium WebDriver (e.g., chromedriver, geckodriver, edgedriver, etc.) in a fully automated manner.

WebDriverManager Maven Dependency

https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager

Selenium Java Maven Dependency

https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java

Selenium Server Maven Dependency

https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-server

TestNG

Test Next Generation Framework, users can use TestNG for test suite execution, batch, group and parallel execution purposes. Also, it has reporting capability.

TestNG Maven Dependency

https://mvnrepository.com/artifact/org.testng/testng

TestNG Annotations

1. @BeforeSuite
2. @BeforeTest
3. @BeforeMethod
4. @Test
5. @AfterMethod
6. @AfterTest
7. @AfterSuite

TestNG Reports

Use Reporter.log to add custom message in the TestNG Report. Examples:

```
Reporter.log("Google search for Iphone 13 done successfully");
```

Attach Screenshot

```
Reporter.log("<a href='" + destFile.getAbsolutePath() + "'> <img src='" +
destFile.getAbsolutePath() + "' height='100' width='100'/> </a>");
```

TestNG_Report_Screenshot_Example.txt

TestNG Assertions

TestNG provides various assertions to validate actual results against expected results. Few examples:

1. assertTrue
    a. Assert.assertTrue(condition)
2. assertFalse
    a. Assert.assertFalse(condition)
3. assertEquals
    a. Assert.assertEquals(actual, expected)
4. assertNotEquals
    a. Assert.assertNotEquals(actual, expected, message)

TestNG Listeners

Listeners are TestNG annotations that literally “listen” to the events in a script and modify TestNG behaviour accordingly. These listeners are applied as interfaces in the code. For example, the most common usage of listeners occurs when taking a screenshot of a particular test that has failed along with the reason for its failure. Listeners also help with logging and generating results.

1. IAnnotationTransformer

2. IAnnotationTransformer2
3. IConfigurable
4. IConfigurationListener
5. IExecutionListener
6. IHookable
7. IInvokedMethodListener
8. IInvokedMethodListener2
9. IMethodInterceptor
10. IReporter
11. ISuiteListener
12. ITestListener

Listener Example:

1. Create a BaseClass with WebDriver object.
2. Create a Java class to automate manual test cases, extend base class to get driver object.
3. Create another Java class for listener **(extend Base class and implement ITestListener to capture screenshots automatically for every failure and attach to Reporter Log)**
4. Add java class listener in TestNG suite xml.
    a. User can add listeners only to java classes also.

Step 1 code:

BaseClass.java

```java
package practice;

import org.openqa.selenium.WebDriver;

public class BaseClass {
    protected static WebDriver driver = null;

}

```

Step 2 code:

DriverExample.java

```java
package practice;

import java.io.File;

public class DriverExample extends BaseClass{
    @BeforeTest()
    public void setup() {
        WebDriverManager.chromedriver().setup();
        driver = new ChromeDriver();
    }
    @Test(priority = 1)
    public void performGoogleSearch() {
        driver.get("https://www.google.com");
        driver.findElement(By.xpath("//input[@name='q']")).sendKeys("Apple Iphone 13" + Keys.ENTER);
        Assert.assertTrue(driver.getTitle().contains("Apple Iphone 14"));
        Reporter.log("Google search for Iphone 13 done successfully");
    }
    @Test(priority = 2)
    public void performWikiSearch() {
        driver.get("https://en.wikipedia.org/wiki/Main_Page");
        driver.findElement(By.xpath("//input[@name='search']")).sendKeys("iphone 13" + Keys.ENTER);
        Reporter.log("Wiki search for Iphone 13 done successfully");
    }
    @AfterTest
    public void tearDown() {
        driver.close();
        driver.quit();
    }
}
```

Step 3 code:

ScreenshotListener.java

```java
package practice;

import java.io.File;

public class ScreenshotListener extends BaseClass implements ITestListener {

    public void onTestFailure(ITestResult result) {
        DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyyMMddHHmmss");
        LocalDateTime now = LocalDateTime.now();
        File scrFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
        try {
            String reportDirectory = new File(System.getProperty("user.dir")).getAbsolutePath()
                    + "/target/surefire-reports";
            File destFile = new File(reportDirectory + "/failure_screenshots/screenshot_" + dtf.format(now) + ".png");
            FileUtils.copyFile(scrFile, destFile);
            Reporter.log("<a href='" + destFile.getAbsolutePath() + "'> <img src='" + destFile.getAbsolutePath()
                    + "' height='100' width='100'/> </a>");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Step 4 Code:

ListenerSuite.xml

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >

<suite name="Suite1">
    <listeners>
        <listener class-name="practice.ScreenshotListener" />
    </listeners>
    <test name="Regression">
        <classes>
            <class name="practice.DriverExample" />
        </classes>
    </test>
</suite>
```

Add Listener only to a specific Java class then use @Listeners annotation. Example:

```
import org.testng.annotations.Listeners;
import org.testng.annotations.Test;

import io.github.bonigarcia.wdm.WebDriverManager;

@Listeners(practice.ScreenshotListener.class)
public class DriverExample extends BaseClass{
```

TestNG Suite XML

Sample TestNG Suite XML:

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >

<suite name="Suite1" verbose="1" >
  <test name="Regression" >
    <classes>
       <class name="practice.DriverExample" />
    </classes>
  </test>
</suite>
```

Refer this link for more TestNG Suite XML options.

TestNG Parallel Execution

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >
<suite name = "Parallel Testing Suite">
   <test name = "Parallel Tests" parallel = "methods">
      <classes>
         <class name = "ParallelTest" />
      </classes>
```

```
   </test>
</suite>
```

Maven TestNG Integration – SureFirePlugin

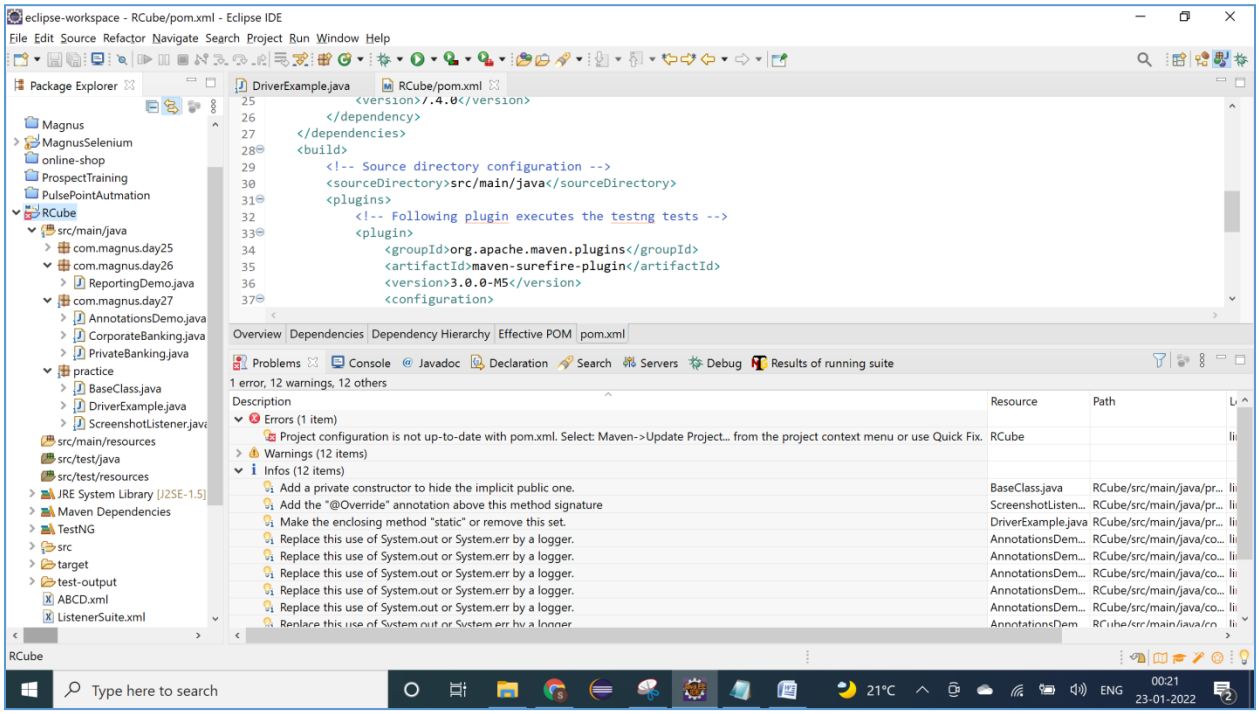
To execute TestNG Suite from Maven then you need SureFirePlugin in the Maven POM.xml.

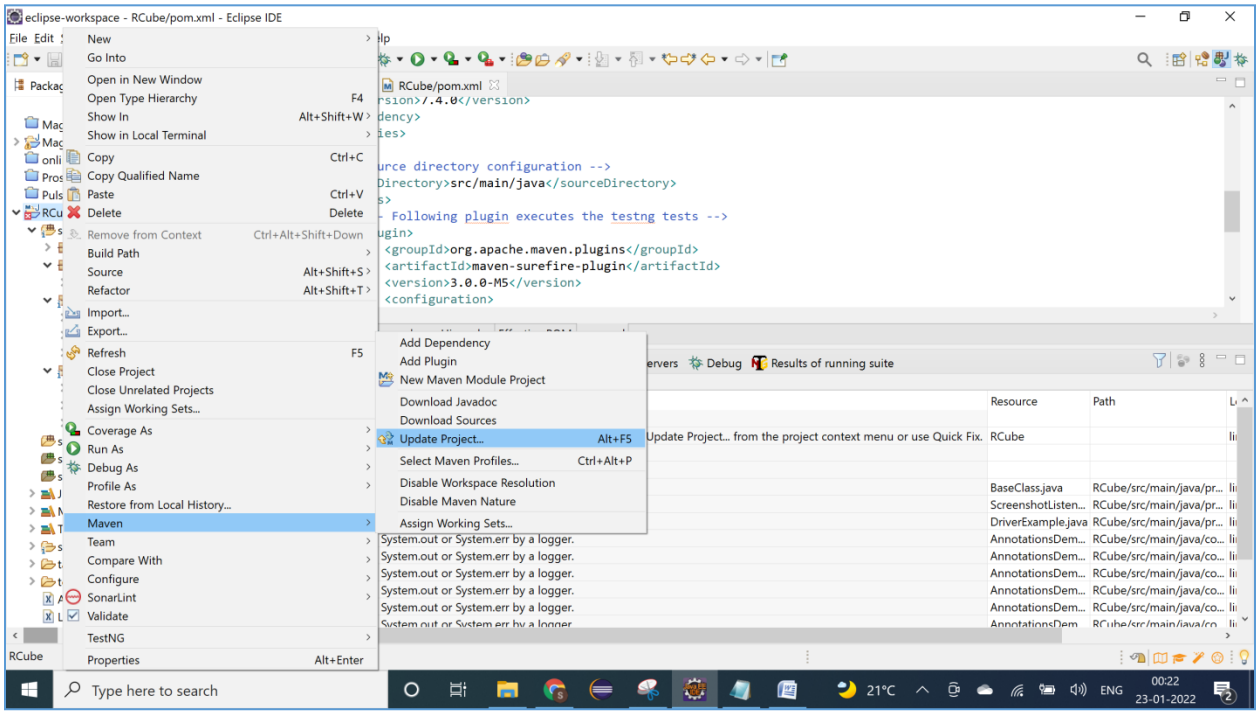
```
27      </dependencies>
28      <build>
29          <!-- Source directory configuration -->
30          <sourceDirectory>src/main/java</sourceDirectory>
31          <plugins>
32              <!-- Following plugin executes the testng tests -->
33              <plugin>
34                  <groupId>org.apache.maven.plugins</groupId>
35                  <artifactId>maven-surefire-plugin</artifactId>
36                  <version>3.0.0-M5</version>
37                  <configuration>
38                      <!-- Suite testng xml file to consider for test execution -->
39                      <suiteXmlFiles>
40                          <suiteXmlFile>ABCD.xml</suiteXmlFile>
41                          <suiteXmlFile>RegressionTests.xml</suiteXmlFile>
42                      </suiteXmlFiles>
43                  </configuration>
44              </plugin>
45              <!-- Compiler plugin configures the java version to be used for compiling the code -->
46              <plugin>
47                  <artifactId>maven-compiler-plugin</artifactId>
48                  <configuration>
49                      <source>1.8</source>
50                      <target>1.8</target>
51                  </configuration>
52              </plugin>
53          </plugins>
54      </build>
55  </project>
```

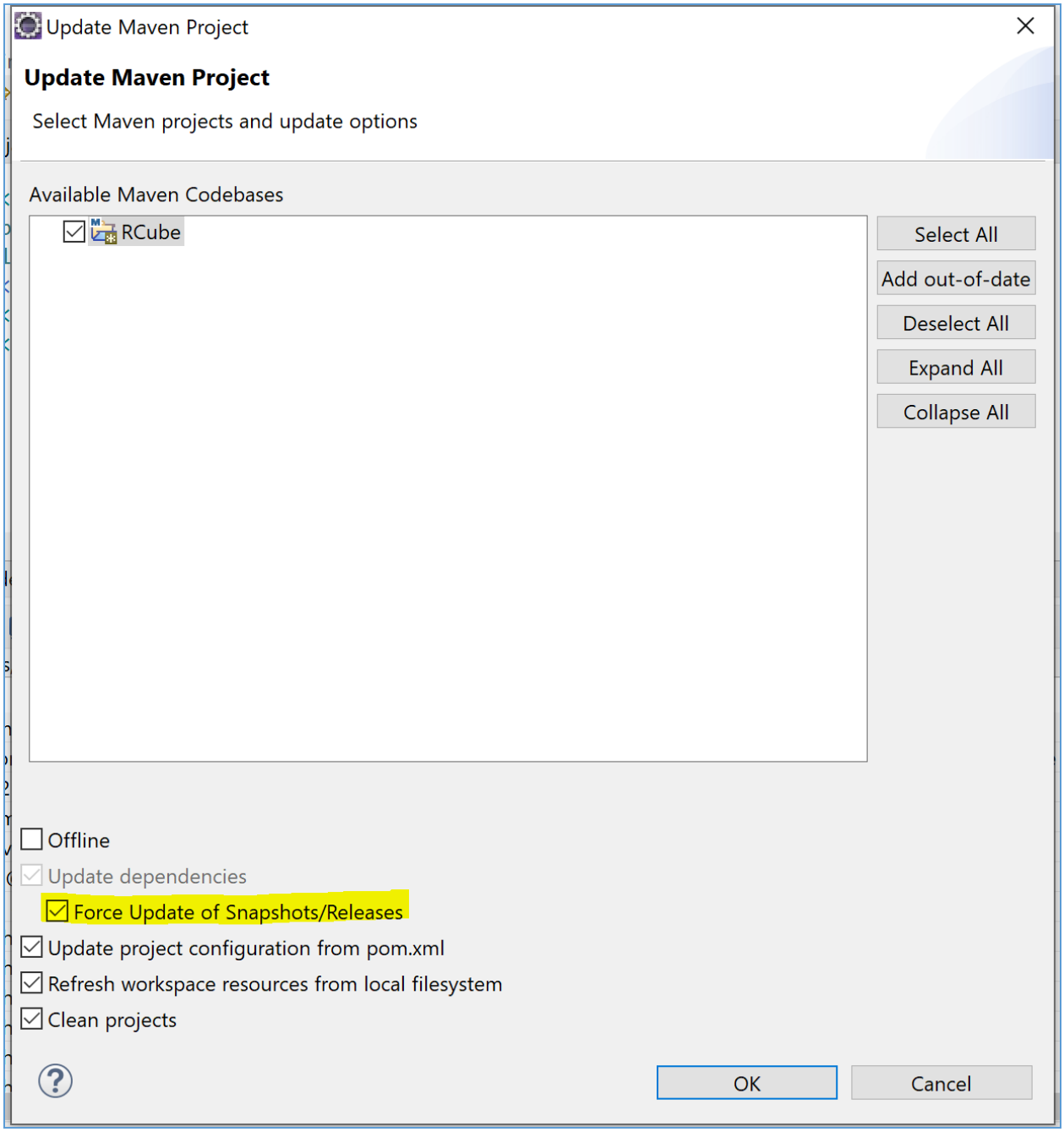
Maven Compiler Plugin and Surefire plugin Code:

```
        <build>
                <!-- Source directory configuration -->
                <sourceDirectory>src/main/java</sourceDirectory>
                <plugins>
                        <!-- Following plugin executes the testng tests -->
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-surefire-plugin</artifactId>
                                <version>3.0.0-M5</version>
                                <configuration>
                                        <!-- Suite testng xml file to consider for test execution -->
                                        <suiteXmlFiles>
                                                <suiteXmlFile>ABCD.xml</suiteXmlFile>
                                                <suiteXmlFile>RegressionTests.xml</suiteXmlFile>
                                        </suiteXmlFiles>
                                </configuration>
                        </plugin>
                        <!--Compiler plugin configures the java version to be used for compiling the code -->
                        <plugin>
                                <artifactId>maven-compiler-plugin</artifactId>
                                <configuration>
                                        <source>1.8</source>
                                        <target>1.8</target>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
```

Maven Force Update Project






Update Maven Project
Update Maven Project
Select Maven projects and update options
Available Maven Codebases
RCube
Select All
Add out-of-date
Deselect All
Expand All
Collapse All
Offline
Update dependencies
Force Update of Snapshots/Releases
Update project configuration from pom.xml
Refresh workspace resources from local filesystem
Clean projects
OK
Cancel

Run Maven Test