Selenium Complete Guide

Emad Nasser

github.com/EmadDNasser/selenium-complete-guide

Contents

Set Up ChromeDriver

```
String chromeKey = "webdriver.chrome.driver";
String chromePath = "D:\\PC\\chromedriver-win64\\chromedriver.exe";
System.setProperty(chromeKey, chromePath);
WebDriver driver = new ChromeDriver();
```

Explanation:

- *System.setProperty()*
  Sets the system property to let Selenium know where your *chromedriver.exe* is located.
- *new ChromeDriver()*
  Launches a new Chrome browser instance.

Tips:

- Always ensure the *ChromeDriver version matches your Chrome browser*.
- Keep the path to the *.exe correct and updated*.
- You can use: *driver.manage().window().maximize();*
  to start the browser in full-screen.

Opens the specified URL in the browser

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

Description: Opens the specified URL in the browser.

Use case: Navigates the WebDriver to the given web page. This is usually the first step in any test case.

Note: The page load will be synchronous, meaning the script will wait until the page is fully loaded.

Finding Elements by XPath and CSS Selector

XPath

```
driver.findElement(By.xpath("//tagname[@attribute='value']"));
```

Description: Finds the first element that matches the given XPath expression.

Use case: Useful when you need precise control over the DOM hierarchy or complex attribute matching.

Example:

```
driver.findElement(By.xpath("//input[@id='username']"));
```

CSS Selector

```
driver.findElement(By.cssSelector("tagname[attribute='value']"));
```

Description: Finds the first element that matches the given CSS selector.

Use case: Preferred for speed and simplicity when the structure is not too complex.

Example:

```
driver.findElement(By.cssSelector("input[id='username']"));
```

Avoid Using By.className with Multiple Class Names

```
// This will throw an error:
driver.findElement(By.className("inputtext _55r1 _6luy _9npi")).sendKeys("Test@example.com");
```

Issue :

Selenium does *not* support compound class names (i.e., class attributes with spaces) in *By.className*.

Error:

```
InvalidSelectorException: Compound class names not permitted
```

Solution :

Use *By.cssSelector* instead:

```
driver.findElement(By.cssSelector(".inputtext._55r1._6luy._9npi")).sendKeys("Test@example.com");
```

Explanation:

In CSS, multiple classes are chained with dots (e.g., *.class1.class2.class3*).

Use XPath to Handle Compound Class Names

```
String Xpath3 = "//*[@class='inputtext _55r1 _6luy _9npi']";
driver.navigate().to("https://www.facebook.com/");
driver.findElement(By.xpath(Xpath3)).sendKeys("Test@example.com");
```

Description:

When an element has multiple classes in its `class` attribute (separated by spaces), `By.className` will not work.

Error you avoid:

```
InvalidSelectorException: Compound class names not permitted
```

Why this works:

XPath treats the whole `class` attribute as a single string, so it can match it exactly—even with spaces.

Tip: To make it more flexible (in case class order changes), you can use `contains`:

```
driver.findElement(By.xpath("//*[contains(@class, 'inputtext')]"));
```

Match Link Text: Full vs. Partial

Match the Whole Link Text

```
driver.get("https://www.linkedin.com/");
driver.findElement(By.linkText("Articles")).click();
```

Description:

Finds a link (*<a>* tag) by ***exact*** visible text.

Use case*:* When you know the full link text and want to click on it directly.

Match Part of the Link Text

```
driver.get("https://www.linkedin.com/login?fromSignIn=true&trk=guest_homepage-basic_nav-header-signin");
driver.findElement(By.partialLinkText("Forgot")).click();
```

Description: Finds a link by matching *part* of its visible text.

Use case: Useful when link text may change slightly or include dynamic content.

Note: Both *linkText* and *partialLinkText* work only for *<a>* elements with visible text.

Use * to Match Any Tag Name in XPath

Match Specific Tag:

```
String Xpath = "//input[@name='ss']";
driver.findElement(By.xpath(Xpath)).sendKeys("Dubai");
```

Description*:*

Matches an element with a specific tag name (*input*) and the *name* attribute equal to *'ss'*.

Match Any Tag (* Wildcard):

```
String Xpath2 = "//*[@name='ss']";
driver.findElement(By.xpath(Xpath2)).sendKeys("Abu Dhabi");
```

Description:

The * wildcard in XPath matches *any tag name* that has the specified attribute.

Use case: Useful when you're not sure of the exact tag or want a more flexible selector.

Page Refresh with Selenium

driver.navigate().refresh();

Description:

Refreshes the current page in the browser.

Use case: When you want to reload the page to check for updated content or reset the state.

Validate XPath in Browser Console

Syntax:

```
$x("XPath")
```

Description:

Evaluates and returns elements matching the XPath expression directly in the browser's developer console (works in Chrome, Firefox, etc.).

Example:

```
$x("//input[@name='ss']")
```

Use case: Quickly test and debug your XPath selectors without running your Selenium script.

Bonus Tip: To test CSS selectors in the console, use:

```
$$("cssSelector")
```

Example:

```
$$("input[name='ss']")
```

Use contains() in XPath to Handle Dynamic Attributes

Syntax:

```
//tagname[contains(@attribute, 'value')]
```

Example:

```
String Xpath4 = "//input[contains(@class,'inputtext _55r1 _6luy _9npi')]";
driver.navigate().to("https://www.facebook.com/");
driver.findElement(By.xpath(Xpath4)).sendKeys("Test@example.com");
```

Description: The *contains()* function allows partial matching of attribute values.

Use case: Great for handling *dynamic IDs or class names* that change between sessions or page loads, especially when they include static substrings.

Bonus: You can also use *contains()* with *text()* to match visible text:

```
//button[contains(text(), 'Login')]
```

Relative vs. Absolute XPath

Relative XPath

```
String XpathRel = "//*[@name='ss']";
driver.findElement(By.xpath(XpathRel)).sendKeys("Abu Dhabi");
```

Description: Starts with *//* and searches the entire DOM for matching elements, regardless of their position.

Use case: *Preferred* for test automation because it's more *flexible* and *resilient* to changes in layout.

Absolute XPath

```
String Xpath5 = "//body/div/div/div/div/div/div/div/div/div/form/div/div/input";
driver.navigate().to("https://www.facebook.com/");
driver.findElement(By.xpath(Xpath5)).sendKeys("admin@example.com");
```

Description: Specifies the *full path* from the root element to the target element.

Use case: Only for quick testing—*fragile* and easily breaks if the DOM structure changes even slightly.

Tip: Always prefer *relative XPath* for stability and maintainability in your automation scripts.

Use XPath Indexing to Target Specific Elements

When multiple elements match the same XPath, you can use an *index* to choose the one you want.

Example 1: Indexing by Position in a Specific Tag

```
String Xpath1 =
"//body/div/div/div/div/div/div/div/div/div/form/div/div/input[2]";
```

Description: Selects the *second* ***<input>*** inside the last * *.

Example 2: Indexing a Group of Matches

```
String Xpath2 = "(//*[@class='inputtext _55r1 _6luy _9npi'])[2]";
```

Description: Finds *all elements* matching the class, and selects the *second match*.

Tips:

- Indexes start from *1* (not 0).
- Use parentheses *()* when indexing a group of elements:

  (`//tagname[@attribute='value'])[index]`

Use #id in CSS Selectors

Syntax 1: With Tag Name

```
driver.get("https://gitlab.com/users/sign_up");
String css_id = "input#new_user_first_name";
driver.findElement(By.cssSelector(css_id)).sendKeys("Emado");
```

Description: Selects an *<input>* element with the ID *new_user_first_name*.

Use case: When you want to be more specific by including the tag name.

Syntax 2: Without Tag Name

```
String css_id2 = "#new_user_first_name";
driver.findElement(By.cssSelector(css_id2)).sendKeys("Emado2");
```

Description: Directly selects the element by its ID using just *#id*.

Use case: When the ID is *unique* on the page (which it should be), this is the *simplest and most efficient* option.

Tip:

- *#id* is *faster* and *more readable* than XPath when the ID is reliable.
- Avoid using it if the *id* is *dynamic* (changes on reload).

Use tagname.classname in CSS Selectors

Syntax 1: With Tag Name

```
String css_id3 = "input.gl-form-input";
driver.navigate().refresh();
driver.findElement(By.cssSelector(css_id3)).sendKeys("Emad3");
```

Description: Selects an *<input>* element with the class *gl-form-input*.

Syntax 2: Using Class Only (Missing)

```
String css_id4 = "gl-form-input"; // This won't work
driver.navigate().refresh();
driver.findElement(By.cssSelector(css_id4)).sendKeys("Emad4");
```

Fix: You must prefix the class with a *dot (.)* in CSS selectors:

```
String css_id4 = ".gl-form-input"; // Correct version
```

Correct Class Selector Without Tag Name

```
String cssClassOnly = ".gl-form-input";
driver.findElement(By.cssSelector(cssClassOnly)).sendKeys("Emad4");
```

Description: Selects *any element* with the class *gl-form-input*, regardless of tag type.

Tips:

- Use *tagname.classname* for more precision.
- Always prefix class names with `.` in CSS selectors.
- Combine multiple classes: `"input.class1.class2"`

Using WebElement in Selenium

Full Example:

```
String chromeKey = "webdriver.chrome.driver";
String chromePath = "D:\\PC\\chromedriver-win64\\chromedriver.exe";
System.setProperty(chromeKey, chromePath);
WebDriver driver = new ChromeDriver();
driver.get("https://www.facebook.com/");
// Declare and use WebElement
WebElement emailButton = driver.findElement(By.id("email"));
emailButton.sendKeys("emad@example.com");
```

Why Use WebElement?

- Improves *readability* and *reusability*
- Ideal when interacting with the *same element multiple times*
- Makes code *cleaner* and easier to maintain

Tip: You can also use *WebElement* for clicks, clear input, checking visibility, etc.:

```
emailButton.clear();
emailButton.click();
boolean isVisible = emailButton.isDisplayed();
```

Static Dropdown in Selenium

Example:

```
String chromeKey = "webdriver.chrome.driver";
String chromePath = "D:\\PC\\chromedriver-win64\\chromedriver.exe";
System.setProperty(chromeKey, chromePath);
WebDriver driver = new ChromeDriver();
driver.navigate().to("https://www.facebook.com/r.php?entry_point=login");
WebElement day = driver.findElement(By.id("day"));
Select dayValue = new Select(day);
// Select different values
dayValue.selectByIndex(5);      // selects 6th item
dayValue.selectByValue("9");     // selects day '9'
dayValue.selectByVisibleText("2");  // selects day '2'
```

Select Class Methods:

- *selectByIndex(int index)* — selects option by index (0-based)
- *selectByValue(String value)* — selects option by the value attribute
- *selectByVisibleText(String text)* — selects option by visible text

Tip: Make sure the *<select>* element is ***not dynamically rendered*** (AJAX/JS), or wait until it's visible using *WebDriverWait*.

Input Dropdown (Auto-suggest Dropdown)

Description: These are treated the same way as regular *<input>* fields. You simply:

1. Send keys to the input box.
2. Wait or interact with the suggestion list (if needed).

Example:

```
String chromeKey = "webdriver.chrome.driver";
String chromePath = "D:\\PC\\chromedriver-win64\\chromedriver.exe";
System.setProperty(chromeKey, chromePath);
WebDriver driver = new ChromeDriver();
driver.get("https://www.booking.com/");
// Input dropdown handled like a regular input field
WebElement locationInput = driver.findElement(By.name("ss"));
locationInput.sendKeys("Dubai");
```

Tips:

- If auto-suggestions appear, use:
  - Arrow keys (*Keys.ARROW_DOWN*) + Enter
  - Or directly locate and click the suggestion with *driver.findElement(By.xpath(...)).click();*
- Use *Thread.sleep()* or ***WebDriverWait*** to give time for suggestions to load.

Waits in Selenium

1. Thread.sleep() *(Java-based | Static)*

```
driver.findElement(By.linkText("Libraries")).click();
Thread.sleep(9000); // Pauses the execution for 9 seconds
driver.findElement(By.linkText("Abc")).click();
```

Not recommended for dynamic pages. Use only when necessary.

2. Implicit Wait *(Selenium | Dynamic | Global)*

```
driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
driver.findElement(By.linkText("Libraries")).click();
driver.findElement(By.linkText("Abc")).click();
```

Applies to all elements globally. Best for ***simple, consistent*** delays.

3. Explicit Wait *(Selenium | Dynamic | Targeted)*

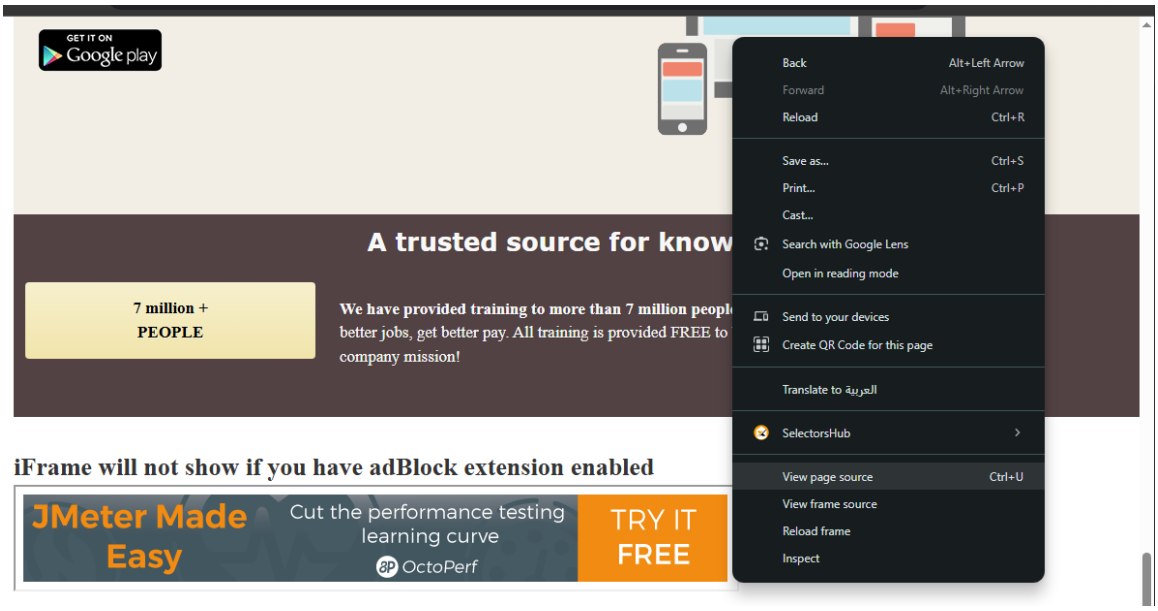
```
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(20)); // Selenium 4+
driver.findElement(By.linkText("Libraries")).click();
wait.until(ExpectedConditions.elementToBeClickable(By.linkText("Abc")));
driver.findElement(By.linkText("Abc")).click();
```

Best for waiting on specific ***conditions or elements***.

Best Practices

- Prefer *Explicit Wait* when dealing with *AJAX elements or dynamic content*.
- Avoid *Thread.sleep()* unless you're debugging or doing timing tests.

Frames in Selenium



Sometimes, elements are *inside an <iframe>*, and you *must switch* to it first. You *can't interact directly* with inner elements unless the focus is switched.

Switch to a Frame

By *ID* or *Name*

driver.switchTo().frame(`"a077aa5e"`*);* `// Switch using frame's id or name`

driver.findElement(By.xpath(`"//img[@src='Jmeter720.png']"`*)).click();*

By WebElement

`WebElement frame` = *driver.findElement(By.xpath(*`"//*[@id='a077aa5e']"`*));*

driver.switchTo().frame(frame);

By Index

```
int count = driver.findElements(By.tagName("iframe")).size(); // Count frames
System.out.println(count); // e.g., 1
driver.switchTo().frame(0); // Switch to first frame by index
WebElement drag = driver.findElement(By.id("draggable"));
WebElement drop = driver.findElement(By.id("droppable"));
Actions action = new Actions(driver);
action.clickAndHold(drag).moveToElement(drop).release().build().perform();
```

Notes on Frames:

- You must *switch into a frame* before interacting with its inner content.
- After you're done, you can return to the *main content* with:

  driver.switchTo().defaultContent();

findElement vs findElements

Method	Behavior on No Match
findElement	❌ Throws *NoSuchElementException*
findElements	✅ Returns an empty list — No exception

```
driver.findElement(By.id("abcd")).click();  // ❌ Throws error if not found
driver.findElements(By.id("abcd"));    // ✅ No error, returns empty list
```

JavaScript Executor in Selenium

JavaScript Executor allows you to run raw JavaScript code inside Selenium. Very helpful for actions like *scrolling*, *clicking hidden elements*, and *highlighting* elements.

Setup:

```
WebElement submitElement = driver.findElement(By.id("philadelphia-field-submit"));
JavascriptExecutor js = (JavascriptExecutor) driver;
```

Common Use Cases:

Scroll to Element

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

Click Element

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

Extract Page Text

```
String pageText = js.executeScript("return document.documentElement.innerText;").toString();
System.out.println(pageText);
```

Show Alert

```
js.executeScript("alert('Hello World');");
```

Draw Border Around Element

```
js.executeScript("arguments[0].style.border='3px solid red'", submitElement);
```

Pro Tip:

Use JavaScript Executor when:

- Standard *click()* fails due to overlays or hidden elements
- You need to interact with dynamic JS-heavy elements
- You want to visually debug with styles or alerts

Locating Elements Using ByAll, ByChained, By.id, By.name

Setup Example

```
String chromeKey = "webdriver.chrome.driver";
String chromePath = "D:\\PC\\chromedriver-win64\\chromedriver.exe";
System.setProperty(chromeKey, chromePath);
WebDriver driver = new ChromeDriver();
driver.get("https://www.facebook.com/");
driver.manage().window().maximize();
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
```

ByAll

Tries ***multiple locators***. Executes the first one that matches.

```
// Try By.id("emad") first, if not found, try By.name("email")
driver.findElement(new ByAll(By.id("emad"), By.name("email"))).sendKeys("emad@example.com");
```

- Good for resilience (if IDs or names might change)
- Waits for up to the implicit timeout before moving to next strategy

ByChained

Used to ***narrow down searches inside parent-child hierarchies***.

```
// Find <div> with id "form-container", then find <input> inside it
WebElement input = driver.findElement(new ByChained(
    By.id("form-container"),
    By.tagName("input")
));
input.sendKeys("text");
```

- Useful for locating deeply nested elements
- Improves accuracy when multiple similar elements exist

By.id, By.name

Simple direct locators:

```
driver.findElement(By.id("email")).sendKeys("emad@example.com");
driver.findElement(By.name("pass")).sendKeys("123456");
```

Tips

- *ByAll* is ***OR*** logic (any one works).
- *ByChained* is ***AND*** logic (must all match in sequence).
- Combine these with *WebDriverWait* for more dynamic behavior.

Set Attribute of Web Element (Using JavaScriptExecutor)

Use Case:

- Pre-fill values into input fields (e.g., date pickers, hidden elements).
- Bypass UI interaction when standard *sendKeys()* fails.

Complete Example:

```
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.JavascriptExecutor;

public class SetAttributeExample {
  public static void main(String[] args) {
    String chromeKey = "webdriver.chrome.driver";
    String chromePath = "D:\\PC\\chromedriver-win64\\chromedriver.exe";
    System.setProperty(chromeKey, chromePath);

    WebDriver driver = new ChromeDriver();

    try {
      // Go to the login page
      driver.get("https://the-internet.herokuapp.com/login");

      // Locate username and password fields
      WebElement usernameField = driver.findElement(By.id("username"));
      WebElement passwordField = driver.findElement(By.id("password"));

      // Use JavaScriptExecutor to set 'value' attributes
      JavascriptExecutor js = (JavascriptExecutor) driver;
      js.executeScript("arguments[0].setAttribute('value', 'tomsmith')", usernameField);
      js.executeScript("arguments[0].setAttribute('value', 'SuperSecretPassword!')",
passwordField);

      // Locate and click login button
      WebElement loginButton = driver.findElement(By.className("radius"));
      loginButton.click();

      // Optional wait
      Thread.sleep(3000);

    } catch (Exception e) {
      e.printStackTrace();
    } finally {
      driver.quit();
    }
  }
}
```

Notes:

- *setAttribute('value', 'yourText')* can be used with any input element (e.g., *<input>*, *<textarea>*, *<select>*).
- This technique is also helpful for ***calendar/date pickers*** when *sendKeys()* doesn't work.
- Always wrap critical actions with ***try-catch-finally*** for safe browser closure.

Step-by-Step: Using WebDriverManager in Selenium Projects

1. Add WebDriverManager Dependency (Maven)

Add the following to your *pom.xml*:

```
<dependency>
  <groupId>io.github.bonigarcia</groupId>
  <artifactId>webdrivermanager</artifactId>
  <version>5.7.0</version> <!-- You can check for the latest version here:
https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->
</dependency>
```

2. Sample Code Using WebDriverManager

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import io.github.bonigarcia.wdm.WebDriverManager;

public class WebDriverManagerExample {
  public static void main(String[] args) {
    // Automatically downloads and sets up the ChromeDriver
    WebDriverManager.chromedriver().setup();

    WebDriver driver = new ChromeDriver();
    driver.get("https://www.google.com");
    System.out.println("Title: " + driver.getTitle());

    driver.quit();
  }
}
```

Advantages:

- *No need to manually download* browser drivers or set system properties.
- Keeps *drivers up to date* automatically.
- Reduces errors related to *driver compatibility*.

Tip: You can also use it for:

```
WebDriverManager.firefoxdriver().setup(); // For Firefox
WebDriverManager.edgedriver().setup();  // For Edge
WebDriverManager.operadriver().setup();  // For Opera
WebDriverManager.chromedriver().browserVersion("114").setup(); // Specific version
```

Print Google Suggestions

```
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.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

import java.time.Duration;
import java.util.List;

public class GoogleSuggestions {
  public static void main(String[] args) {
    WebDriver driver = new ChromeDriver();
    driver.manage().window().maximize();

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

      // Accept cookies if prompted (optional based on region)
      try {
        WebElement acceptBtn = driver.findElement(By.xpath("//div[contains(text(),'I
agree')]"));
        acceptBtn.click();
      } catch (Exception e) {
        // No cookie prompt
      }

      driver.findElement(By.name("q")).sendKeys("Toyota");

      String ulXpath = "//ul[@jsname='bw4e9b']";

      WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
      WebElement list = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(ulXpath)));

      List<WebElement> liList = list.findElements(By.tagName("span"));

      for (WebElement li : liList) {
        String suggestion = li.getText();
        if (!suggestion.isEmpty()) {
          System.out.println(suggestion);
        }
      }

      System.out.println("----END----");

    } catch (Exception e) {
      e.printStackTrace();
    } finally {
      driver.quit();
    }
```

```
  }
}
```

Key Notes:

- *Cookie popup handling* is added for completeness.
- *for-each* loop improves readability.
- Ensures browser closes in the *finally* block.

Handle SSL & Use Command-Line Switches

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.remote.CapabilityType;

public class SSLandChromeSwitches {
  public static void main(String[] args) {

    // Setup ChromeOptions to handle SSL certificates and use incognito mode
    ChromeOptions options = new ChromeOptions();
    options.setCapability(CapabilityType.ACCEPT_INSECURE_CERTS, true);
    options.addArguments("--incognito"); // Opens browser in incognito mode

    WebDriver driver = new ChromeDriver(options);
    driver.manage().window().maximize();
    driver.get("http://google.com");

    // Optionally, close the browser
    driver.quit();
  }
}
```

What It Does:

- *ACCEPT_INSECURE_CERTS* → Accepts self-signed or expired SSL certificates (useful in test environments).
- *--incognito* → Launches the browser in incognito mode, avoiding cache/cookies issues.

Additional Useful Switches (for future use):

You can add more arguments like:

```
options.addArguments("--start-maximized");
options.addArguments("--disable-popup-blocking");
options.addArguments("--disable-extensions");
```

Full list: peter.sh Chromium Switches

Click Element by Text (with text() and contains())

```
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.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import java.time.Duration;

public class ClickByText {
  public static void main(String[] args) {

    WebDriver driver = new ChromeDriver();
    driver.manage().window().maximize();
    driver.manage().deleteAllCookies();

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

    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));

    // Option 1: Exact match using text()
    WebElement element = wait.until(ExpectedConditions.elementToBeClickable(
        By.xpath("//*[text()='News']")));
    element.click();

    // Option 2: Use contains in case of spaces or dynamic text
    // WebElement element =
wait.until(ExpectedConditions.elementToBeClickable(
    //         By.xpath("//*[contains(text(),'News')]")));
    // element.click();
  }
}
```

Get Attribute:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import io.github.bonigarcia.wdm.WebDriverManager;

public class GetAttributeExample {
  public static void main(String[] args) {

    WebDriverManager.chromedriver().setup();
    WebDriver driver = new ChromeDriver();
```

```
    driver.get("https://github.com/login");
    driver.manage().window().maximize();

    // Locate the <a> tag that wraps the GitHub logo (update selector as
needed)
    WebElement githubIconLink = driver.findElement(By.xpath("//a[contains(@href,
'github.com')]"));

    // Get the href attribute value
    String hrefValue = githubIconLink.getAttribute("href");
    System.out.println("GitHub icon link: " + hrefValue);

    // Navigate to that URL
    driver.get(hrefValue);
  }
}
```

Notes:

- You used *getAttribute("href")* correctly — that's how to extract attribute values from elements.
- Make sure you select an element that actually has the *href* attribute (like *<a>*).
- *WebDriverManager* ensures the driver is compatible and properly downloaded — good use

Notes:

- *text()='News'* works only when the text matches ***exactly*** with no leading/trailing whitespace.
- *contains(text(),'News')* is ***more flexible*** and helps when extra whitespace or characters are present.
- Your comment on white space and using *contains* is 💯 spot-on.

Framework:

In *Selenium*, a *framework* is a structured and reusable set of guidelines, tools, and best practices that help in designing, developing, and maintaining *automated test scripts* effectively and efficiently.

What a Framework in Selenium Includes:

Component	Description
Folder structure	Organizes test cases, utilities, reports, drivers, etc.
Reusable code	Common functions like login, setup, teardown, reporting, etc.
Data handling	Uses external files (like Excel, CSV, JSON, DB) to feed test data.
Reporting	Generates test result reports (e.g., ExtentReports, Allure).
Assertions	Verifies actual vs expected results (e.g., TestNG/JUnit assertions).
CI/CD support	Integrated with tools like Jenkins or GitLab CI for automation.

Common Types of Selenium Frameworks:

Type	Description
Modular	Divides test cases into small, independent modules.
Data-Driven	Uses external data sources (Excel, JSON, DB) to drive test cases.
Keyword-Driven	Test steps (keywords) are written in an external file and mapped to functions.
Hybrid	Combines features of data-driven and keyword-driven frameworks.
Behavior-Driven (BDD)	Uses natural language for test cases (with Cucumber + Gherkin syntax).

Common Tools and Technologies Used:

- *Testing libraries*: TestNG, JUnit
- *Build tools*: Maven, Gradle
- *Reporting*: ExtentReports, Allure
- *CI tools*: Jenkins, GitLab CI, GitHub Actions
- *Logging*: Log4j, SLF4J
- *Dependency management*: WebDriverManager

Example Folder Structure:

```
selenium-framework/
│
├── src/
│   ├── test/java/
│   │   ├── tests/
│   │   ├── pages/
│   │   ├── utilities/
│   │   └── testdata/
│
├── testng.xml
├── pom.xml (if using Maven)
└── reports/
```

Why Use a Framework?

- Increases *code reusability*
- Simplifies *test maintenance*
- Improves *scalability and reliability*
- Enables *continuous testing* with CI/CD
- Makes test scripts more *readable and organized*

TestNG

TestNG (Test Next Generation) is a *testing framework* inspired by JUnit and NUnit, but with *more powerful features* — especially for automation with *Selenium WebDriver*.

Why Use TestNG with Selenium?

Feature	Benefit
Annotations	Controls test flow easily (*@Test*, *@BeforeMethod*, etc.)
Test Configuration	Set dependencies, priorities, groups, etc.
Data-driven testing	Built-in support using *@DataProvider*
Parallel execution	Run tests concurrently to save time
Reports	Generates default HTML/XML test reports
Integration	Works well with tools like Maven, Jenkins, Allure, etc.

Common Annotations in TestNG:

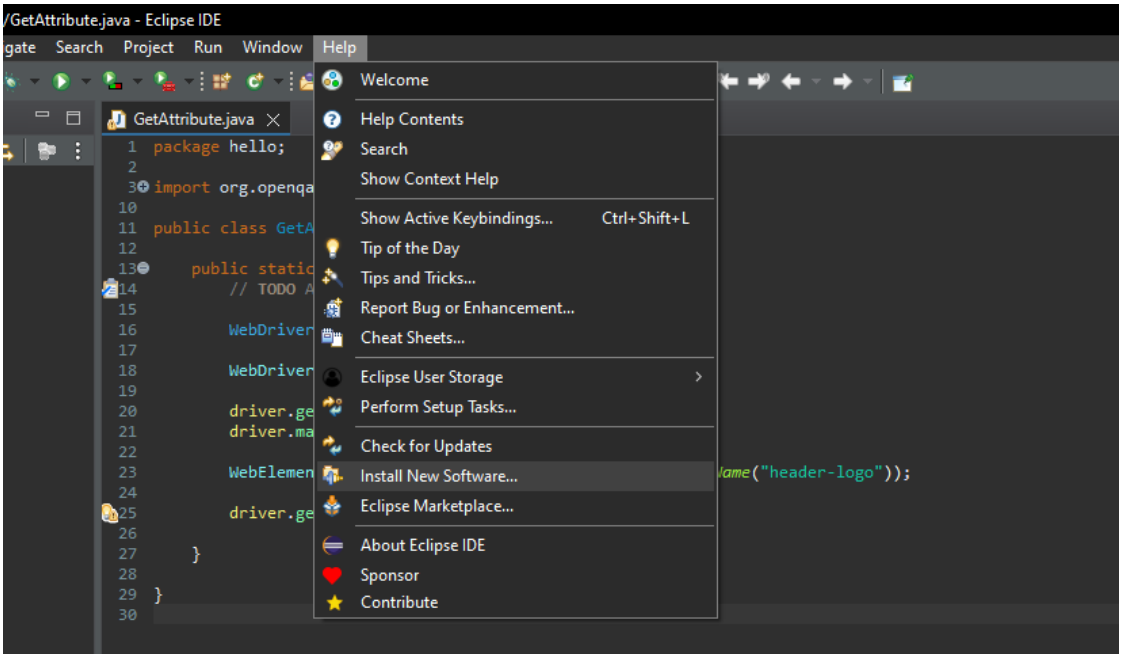
Annotation	Description
@Test	Marks a method as a test
@BeforeMethod / *@AfterMethod*	Run before/after each test method
@BeforeClass / *@AfterClass*	Run once before/after all methods in the class
@DataProvider	Supplies data for data-driven tests
@Parameters	Used for parameterized tests from *testng.xml*
@BeforeSuite / *@AfterSuite*	Executes before/after the whole suite

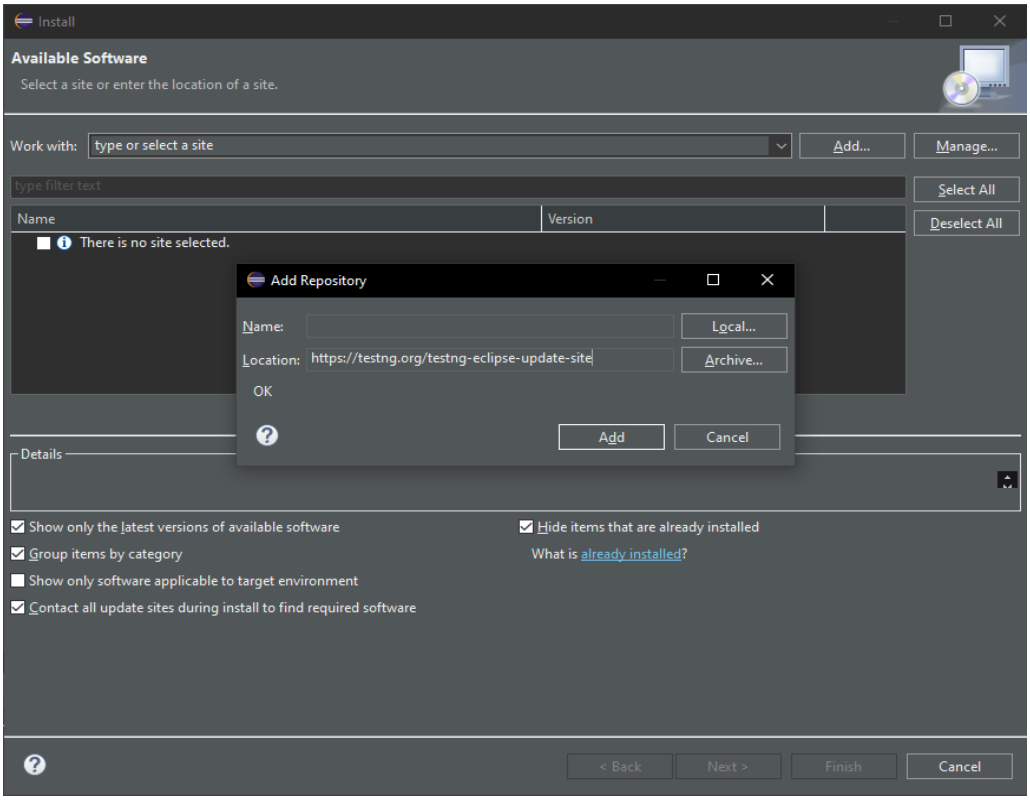
Add TestNG to Eclipse:

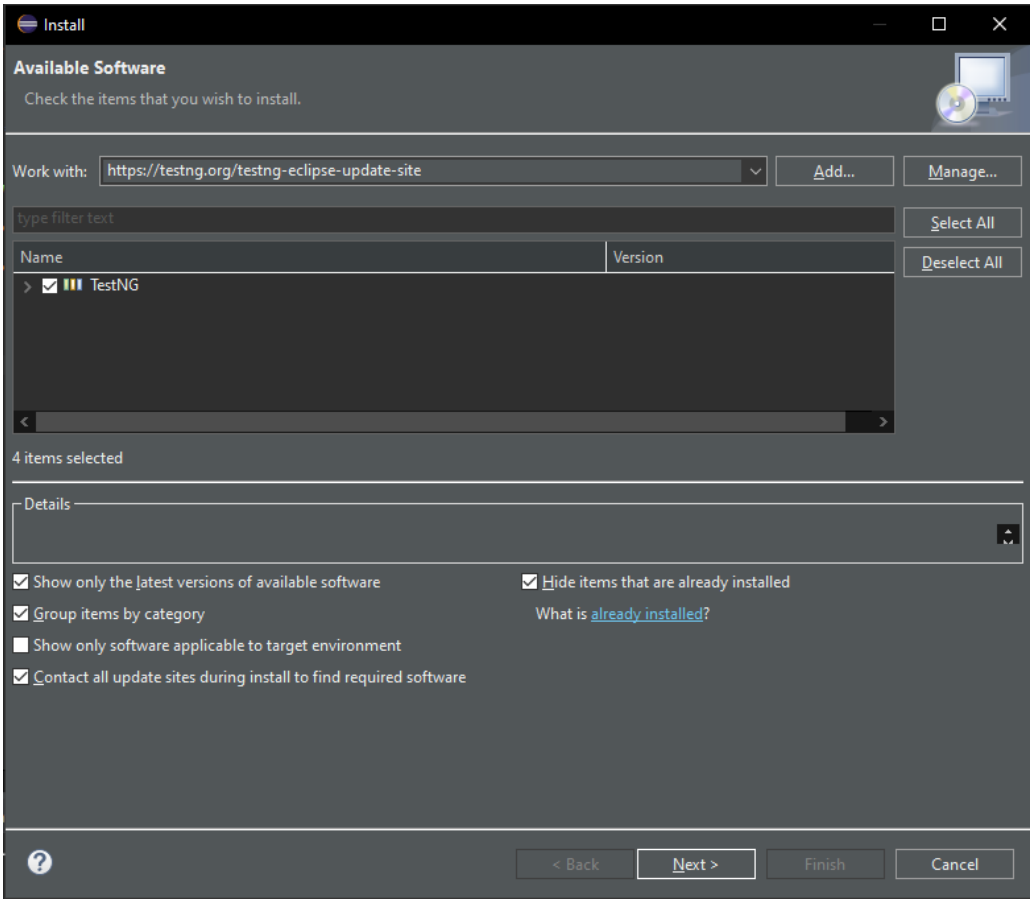
https://testng.org/testng-eclipse/download

Install from update site

- Select Help / Install New Software…
- Enter the update site URL in “Work with:” field:
    - Update site for release: `https://testng.org/testng-eclipse-update-site`
- Make sure the check box next to URL is checked and click Next.
- Eclipse will then guide you through the process.




Install
Available Software
Select a site or enter the location of a site.
Work with:
type or select a site
Add...
Manage...
type filter text
Select All
Deselect All
Name
Version
There is no site selected.
Add Repository
Name:
Local...
Location:
https://testng.org/testng-eclipse-update-site
Archive...
OK
Add
Cancel
Details
Show only the latest versions of available software
Group items by category
Show only software applicable to target environment
Contact all update sites during install to find required software
Hide items that are already installed
What is already installed?
< Back
Next >
Finish
Cancel


Install
Available Software
Check the items that you wish to install.
Work with:
https://testng.org/testng-eclipse-update-site
Add...
Manage...
type filter text
Select All
Deselect All
Name
Version
TestNG
4 items selected
Details
Show only the latest versions of available software
Group items by category
Show only software applicable to target environment
Contact all update sites during install to find required software
Hide items that are already installed
What is already installed?
< Back
Next >
Finish
Cancel

Create Test cases using TestNG:

```
package com.freecrm.testcases;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;

public class LoginPageTest {

	@Test
	public void titleTest() {
		String ChromeDriverPath = "D:\\PC\\chromedriver-
win64\\chromedriver.exe";
		String chromeDriverKey = "webdriver.chrome.driver";
		System.setProperty(chromeDriverKey, ChromeDriverPath);
		WebDriver driver = new ChromeDriver();
		String freecrmURL = "https://ui.cogmento.com/";
		driver.get(freecrmURL);

		String ExpectedResultString = "Cogmento CRM";
		String ActualResultString = driver.getTitle();
		System.out.println(ExpectedResultString.equals
				(ActualResultString));
		//driver.quit();
	}

	@Test
	public void urlCheck() {
		String ChromeDriverPath = "D:\\PC\\chromedriver-
win64\\chromedriver.exe";
		String chromeDriverKey = "webdriver.chrome.driver";
		System.setProperty(chromeDriverKey, ChromeDriverPath);
		WebDriver driver = new ChromeDriver();
		String freecrmURL = "https://ui.cogmento.com/";
		driver.get(freecrmURL);

		String ExpectedResultString = "https://ui.cogmento.com/";
		String ActualResultString = driver.getCurrentUrl();
		System.out.println(ExpectedResultString.equals
				(ActualResultString));

		//driver.quit();
	}

	@Test
```

```
	public void logoTest() {
		String ChromeDriverPath = "D:\\PC\\chromedriver-
win64\\chromedriver.exe";
		String chromeDriverKey = "webdriver.chrome.driver";
		System.setProperty(chromeDriverKey, ChromeDriverPath);
		WebDriver driver = new ChromeDriver();
		String freecrmURL ="https://classic.freecrm.com/index.html";
		driver.get(freecrmURL);

		WebElement logoElement =
driver.findElement(By.xpath("//img[@src='https://classic.freecrm.com/i
mg/logo.png']"));
		boolean ExpectedResultBoolean = true;
		boolean ActualResultBoolean = logoElement.isDisplayed();
		System.out.println(ExpectedResultBoolean ==
				ActualResultBoolean);
		//driver.quit();
	}

	@Test
	public void loginTest() {
		String ChromeDriverPath = "D:\\PC\\chromedriver-
win64\\chromedriver.exe";
		String chromeDriverKey = "webdriver.chrome.driver";
		System.setProperty(chromeDriverKey, ChromeDriverPath);
		WebDriver driver = new ChromeDriver();
		String freecrmURL = "https://ui.cogmento.com/";
		driver.get(freecrmURL);
		driver.manage().window().maximize();

		WebElement usernamElement =
			driver.findElement(By.name("email"));
		WebElement passwordElement =
			driver.findElement(By.name("password"));
		WebElement buttonElement =
driver.findElement(By.cssSelector("#ui > div > div > form > div >
div.ui.fluid.large.blue.submit.button"));
		usernamElement.sendKeys("emad.naser1@gmail.com");
		passwordElement.sendKeys("123456");
		buttonElement.submit();
		//driver.quit();
	}
}
```

Priorities test cases

```
@Test (priority = 1)
      public void titleTest() {

....
}
```

Before and After Method:

The ***BeforeMethod*** and the **AfterMethod** are executed every time the Java methods execute (`titleTest(), urlCheck(), logoTest(), loginTest()..etc)`

```
public class LoginPageTest {

WebDriver driver;

      @BeforeMethod
      public void setUP() {
            String ChromeDriverPath = "D:\\PC\\chromedriver-
win64\\chromedriver.exe";
            String chromeDriverKey = "webdriver.chrome.driver";
            System.setProperty(chromeDriverKey, ChromeDriverPath);
            driver = new ChromeDriver();
      }

      @AfterMethod
      public void tearDown() {
            driver.quit();
      }


@Test (priority = 1)
      public void titleTest() {

            String freecrmURL = "https://ui.cogmento.com/";
            driver.get(freecrmURL);

.....

}
```

<u>Note</u>: it's not necessary to locate those two methods at the beginning of the class

Assertion:

Hard Assertion:

Stops the test if the condition fails.

A *hard assertion* is a type of *assertion* used in testing that, when it fails, *immediately stops the execution* of the test case or test method. It's often used when the subsequent steps depend on the success of the assertion.

```
public void loginTest() {

            String freecrmURL = "https://ui.cogmento.com/";
            driver.get(freecrmURL);
            driver.manage().window().maximize();
            WebElement usernameElement =
driver.findElement(By.name("email"));
            WebElement passwordElement =
driver.findElement(By.name("password"));
            WebElement buttonElement =
driver.findElement(By.cssSelector("#ui > div > div > form > div >
div.ui.fluid.large.blue.submit.button"));
            usernameElement.sendKeys("emad.naser1@gmail.com");
            passwordElement.sendKeys("123456");
            buttonElement.click();
        WebDriverWait wait = new WebDriverWait(driver,
Duration.ofSeconds(10));
        WebElement userDisplayElement =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelecto
r(".user-display")));
            String ActualResultString =  userDisplayElement.getText();

            String ExpectedResultString = "Emad Nasser"; // Pass
            //String ExpectedResultString = "Emad "; // Output error:
expected [Emad ] but found [Emad Nasser]

System.out.println(ActualResultString.equals(ExpectedResultString));
      Assert.assertEquals(ActualResultString, ExpectedResultString, "Username or
password are not correct! ");
      }

assertFalse(false); // False expected false => Output: pass
assertFalse(true); //  False expected true => Output: false
assertTrue(false); //  True expected false => Output: false
assertTrue(false); //  True expected false => Output : false
Assert.assertNotEquals("Emad", "Emad","The result shoul not be equal");
```

Soft Assertion:

Logs the failure but continues running the test.

A *soft assertion* allows the test to **continue execution** even if an assertion fails. Unlike hard assertions, soft assertions don't immediately stop the test; instead, they collect all failures and report them at the end of the test.

```
@Test
      public void titleTest1() {

            SoftAssert softAssert = new SoftAssert();
            String freecrmURL = "https://ui.cogmento.com/";
            driver.get(freecrmURL);
            String ExpectedResultString = "Cogmento";
            String ActualResultString = driver.getTitle();

           System.out.println(ExpectedResultString.equals(
                              ActualResultString));// Output: true
            softAssert.assertEquals(ActualResultString,
                                    ExpectedResultString);

            softAssert.assertAll(); // You must add assertAll(); in order to
print Pass  / Fail, otherwise will print pass even though the test is fail.
            //driver.quit();
            System.out.println("This line was printed even though the
output is false");

      }
```

<u>Note:</u>

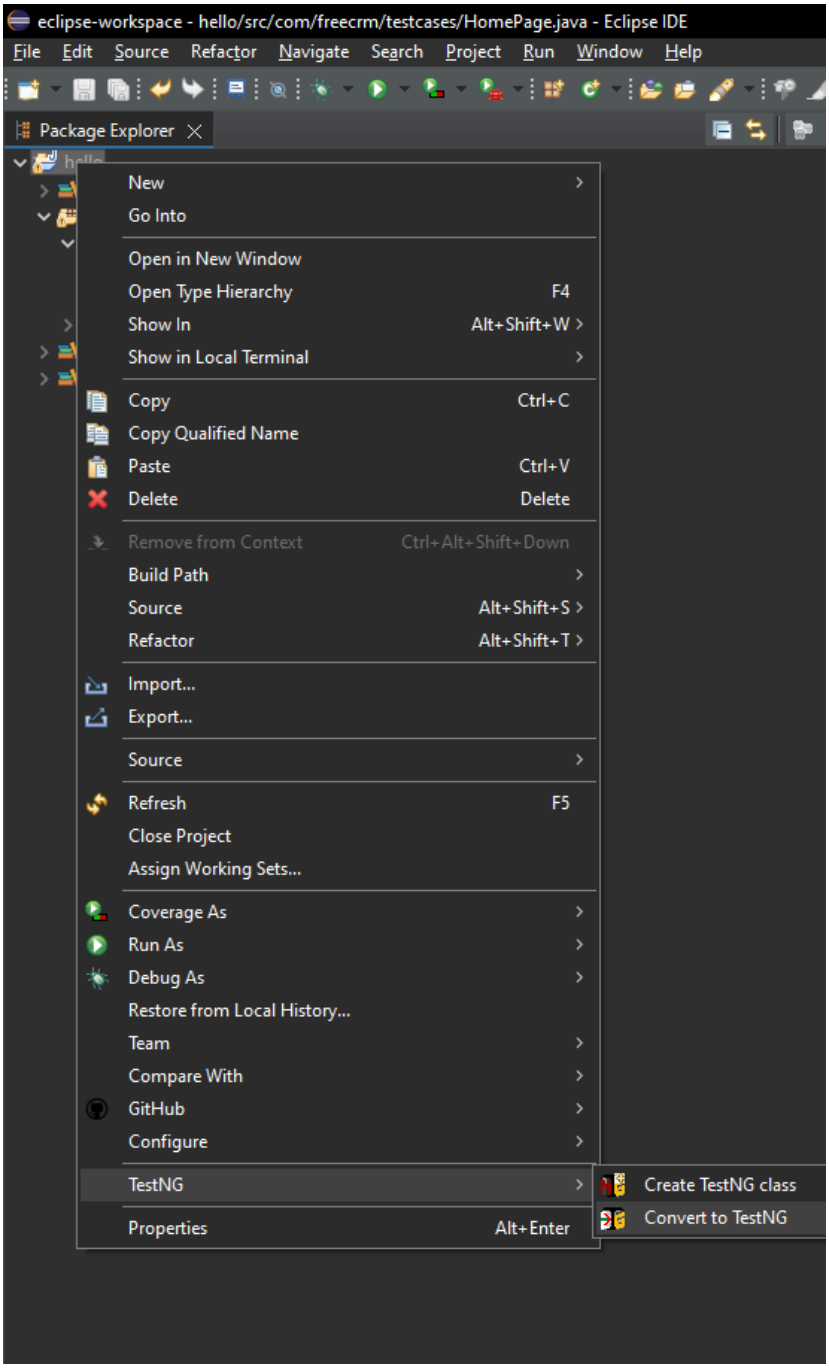
- **You must add assertAll(); in order to print Pass / Fail, otherwise will print pass even though the test is fail.**
- **Lines after assertAll(); will be executed in case the test is fail, and won't be executed in case the test is pass.**

TestNG XML File:

Right click on the project => **TestNG** => **Convert to TestNG**

Fill “**Suit Name**” and “**Test Name**”, then click “**Finish**”

An XML file will be created “**testing.xml**”



Refactoring

Generate testng.xml

☑ Generate testng.xml

Location: /hello/testng.xml Browse...

Suite name: hello test

Test name: End to End Test

Class selection: Classes Parallel mode: none Thread count:

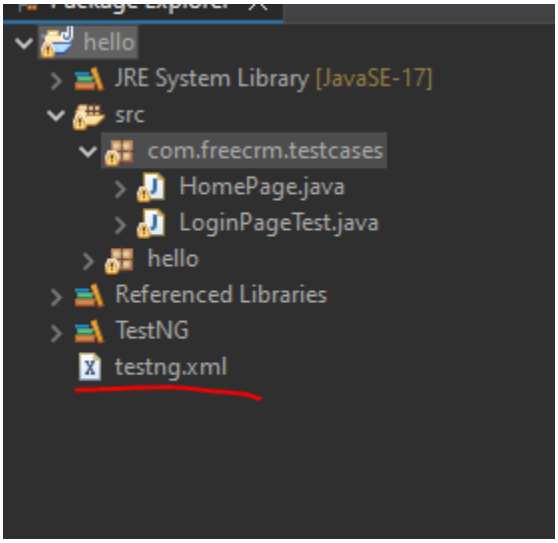
Preview

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="hello test">
  <test thread-count="5" name="End to End Test">
    <classes>
      <class name="com.freecrm.testcases.HomePage"/>
      <class name="com.freecrm.testcases.LoginPageTest"/>
    </classes>
  </test> <!-- End to End Test -->
</suite> <!-- hello test -->
```

Code generation

suite() methods: Remove

< Back | Next > | Finish | Cancel



Include and Exclude Methods TestNG XML File:

When you run this TestNG suite:

- The **`HomePage`** class will run all its test methods except **`clickOnDealsTest`**.
- The **`LoginPageTest`** class will run **only** the **`urlCheck`** method.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="hello test">
  <test thread-count="5" name="End to End Test">
    <classes>
      <class name="com.freecrm.testcases.HomePage">
        <methods>
        <exclude name="clickOnDealsTest"></exclude>
      </methods>
       </class>
      <class name="com.freecrm.testcases.LoginPageTest">
       <methods>
         <include name="urlCheck"></include>
       </methods>
      </class>
    </classes>
  </test> <!-- End to End Test -->
</suite> <!-- hello test -->
```

BeforeClass and AfterClass:

are TestNG annotations used to define setup and teardown methods *that run once per class* — before and after all test methods in that class, respectively.

Annotation	Runs Before/After	Frequency	Common Use Case
`@BeforeClass`	Test methods in a **class**	**Once per class**	Launch browser, set up WebDriver
`@AfterClass`	Test methods in a **class**	**Once per class**	Quit browser, clean up resources
`@BeforeMethod`	**Each** `@Test` method	**Before every test method**	Log in, reset state, prepare test data
`@AfterMethod`	**Each** `@Test` method	**After every test method**	Log out, reset data, take screenshot

- `@BeforeClass` + `@AfterClass` → One browser per **test class**.
- `@BeforeMethod` + `@AfterMethod` → One browser per **test method**.

Example:

```java
public class ExampleTest {

    @BeforeClass
    public void beforeClass() {
        System.out.println("Launch browser - runs ONCE");
    }

    @BeforeMethod
    public void beforeMethod() {
        System.out.println("Log in - runs BEFORE EVERY TEST");
    }

    @Test
    public void test1() {
        System.out.println("Running test 1");
    }

    @Test
    public void test2() {
        System.out.println("Running test 2");
    }

    @AfterMethod
    public void afterMethod() {
        System.out.println("Log out - runs AFTER EVERY TEST");
    }

    @AfterClass
    public void afterClass() {
        System.out.println("Close browser - runs ONCE");
    }
}
```

Before and after test and suit:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="hello test">
 <test thread-count="5" name="End to End Test">
  <classes>
   <class name="com.freecrm.testcases.HomePage"></class>
   <class name="com.freecrm.testcases.LoginPageTest"></class>
  </classes>
 </test> <!-- End to End Test -->

 <test thread-count="5" name="Sanity Test">
  <classes>
   <class name="com.freecrm.testcases.HomePage"></class>
  </classes>
 </test> <!-- End to End Test -->

 <test thread-count="5" name="Regression Test">
  <classes>
   <class name="com.freecrm.testcases.LoginPageTest"></class>
  </classes>
  </test> <!-- End to End Test -->

</suite> <!-- hello test -->
```

The differences:

```
public class testNG {

      // suite
        // Multi tests
         // Multi classes
          // Multi Methods (Test Cases).

      @BeforeSuite
      public void beforeSuite() {
            System.out.println("I will execute before suite (Before all
the tests, classes and methods)");
      }

      @AfterSuite
      public void afterSuite() {
            System.out.println("I will execute after suite (After all
the tests, classes and methods)");
      }

      @BeforeTest
      public void beforeTest() {
            System.out.println("I will execute before test (After all
the classes and methods)");
      }

      @AfterTest
      public void afterTest() {
            System.out.println("I will execute after test (After all the
classes and methods");
      }

      @BeforeClass
      public void beforeClass() {
            System.out.println("I will execute before this class (Before
all the methods");
      }

      @AfterClass
      public void afterClass() {
            System.out.println("I will execute After this class (After
all the methods");
      }

      @BeforeMethod
```

```
	public void beforeMethod() {
		System.out.println("I will execute before method (Before any
methods in this class");
	}

	@AfterMethod
	public void afterMethod() {
		System.out.println("I will execute after method (After every
methods in this class");
	}

	@Test
	public void testCase1() {
		System.out.println("This is the first test case");
	}

	@Test
	public void testCase2() {
		System.out.println("This is the second test case");
	}

	@Test
	public void testCase3() {
		System.out.println("This is the third test case");
	}

	@Test
	public void testCase4() {
		System.out.println("This is the fourth test case");
	}
}
```

Output:

```
I will execute before test (After all the classes and methods)
I will execute before this class (Before all the methods)
I will execute before method (Before any methods in this class)
This is the first test case
I will execute after method (After every methods in this class)
I will execute before method (Before any methods in this class)
This is the second test case
I will execute after method (After every methods in this class)
I will execute before method (Before any methods in this class)
This is the third test case
I will execute after method (After every methods in this class)
I will execute before method (Before any methods in this class)
This is the fourth test case
I will execute after method (After every methods in this class)
I will execute After this class (After all the methods)
I will execute after test (After all the classes and methods)
PASSED: com.freecrm.testcases.testNG.testCase1
PASSED: com.freecrm.testcases.testNG.testCase2
PASSED: com.freecrm.testcases.testNG.testCase4
PASSED: com.freecrm.testcases.testNG.testCase3

===============================================
    Default test
    Tests run: 4, Failures: 0, Skips: 0
===============================================

I will execute after suite (After all the tests, classes and methods)

===============================================
Default suite
Total tests run: 4, Passes: 4, Failures: 0, Skips: 0
```

Group Test

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="hello test">
 <test thread-count="5" name="End to End Test">
  <groups>
   <run>
     <include name ="Sanity"></include>
   </run>
  </groups>
  <classes>
   <class name="com.freecrm.testcases.HomePage"></class>
   <class name="com.freecrm.testcases.LoginPageTest"></class>
  </classes>
 </test> <!-- End to End Test -->
 </suite> <!-- hello test -->
```

```
public class HomePage {


      // E2E: 2
      // Sanity: 1
      // Regression: 1

      // Total:
      // E2E: 6
      // Sanity: 2
      // Regression: 2

      WebDriver driver;

      @BeforeMethod (groups =  {"Sanity","E2E","Regression"})
      public void login() {
      String ChromeDriverPath = "D:\\PC\\chromedriver-
win64\\chromedriver.exe";
      String chromeDriverKey = "webdriver.chrome.driver";
      System.setProperty(chromeDriverKey, ChromeDriverPath);
      driver = new ChromeDriver();
      driver.get("https://ui.cogmento.com/home");
      driver.manage().window().maximize();
      driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10))
;
      WebElement usernameElement =
driver.findElement(By.name("email"));
```

```
      WebElement passwordElement =
driver.findElement(By.name("password"));
      WebElement buttonElement = driver.findElement(By.cssSelector("#ui
> div > div > form > div > div.ui.fluid.large.blue.submit.button"));
      usernameElement.sendKeys("emad.naser1@gmail.com");
      passwordElement.sendKeys("123456");
      buttonElement.click();

      }

      @AfterMethod (groups =  {"Sanity","E2E","Regression"})
      public void tearDown() {
            driver.quit();
      }

      @Test (priority = 5, groups = {"E2E", "Sanity"})
      public void clickOnContactTest() {
            WebElement contactsElement =
driver.findElement(By.xpath("//span[text()='Contacts']"));
            contactsElement.click();
            WebElement statusLablElement =
driver.findElement(By.xpath("//th[text()='Status']"));
            boolean ActualResultBooloean =
statusLablElement.isDisplayed();
            assertTrue(ActualResultBooloean, "The status lable text not
displayed");

      }
      @Test (priority = 6, groups = {"Regression"})
      public void clickOnDealsTest() {
            WebElement contactsElement =
driver.findElement(By.xpath("//span[text()='Deals']"));
            contactsElement.click();
            WebElement statusLablElement =
driver.findElement(By.xpath("//th[text()='Status']"));
            boolean ActualResultBooloean =
statusLablElement.isDisplayed();
            assertTrue(ActualResultBooloean, "The status lable text not
displayed");

      }




      @Test (priority = 7, groups = {"E2E"})
      public void clickOnTasksTest() {
```

```
            WebElement contactsElement =
driver.findElement(By.xpath("//span[text()='Tasks']"));
            contactsElement.click();
            WebElement statusLablElement =
driver.findElement(By.xpath("//th[text()='Status']"));
            boolean ActualResultBooloean =
statusLablElement.isDisplayed();
            assertTrue(ActualResultBooloean, "The status lable text not
displayed");

      }

}
```

```
public class LoginPageTest {

      // E2E: 4
      // Sanity: 1
      // Regression: 1

      // Total LoginPageTest + HomePage:
      // E2E: 6
      // Sanity: 2
      // Regression: 2

      WebDriver driver;

      @Test (priority = 1, groups = {"E2E"})
      public void titleTest() {

            String freecrmURL = "https://ui.cogmento.com/";
            driver.get(freecrmURL);
            String ExpectedResultString = "Cogmento CRM";
            String ActualResultString = driver.getTitle();
             System.out.println(ExpectedResultString.equals(
                                    ActualResultString));
            // Hard assertion
            Assert.assertEquals(ActualResultString,
                                    ExpectedResultString);

      }

      @Test (priority = 2, groups = {"E2E"})
      public void urlCheck() {

            String freecrmURL = "https://ui.cogmento.com/";
            driver.get(freecrmURL);
```

```
            String ExpectedResultString = "https://ui.cogmento.com/";
            String ActualResultString = driver.getCurrentUrl();
            System.out.println(ExpectedResultString.equals(
                                ActualResultString));
            Assert.assertEquals(ActualResultString,
                                 ExpectedResultString);
            //driver.quit();
      }

      @Test (priority = 3, groups =  {"E2E"})
      public void logoTest() {

            String freecrmURL = https://classic.freecrm.com/index.html";
            driver.get(freecrmURL);
            WebElement logoElement = driver.findElement(By.xpath(
 "//img[@src='https://classic.freecrm.com/img/logo.png']"));
            boolean ExpectedResultBoolean = true;
            boolean ActualResultBoolean = logoElement.isDisplayed();

            System.out.println(ExpectedResultBoolean ==
                                ActualResultBoolean);
            Assert.assertEquals(ActualResultBoolean,
                                ExpectedResultBoolean);
            //driver.quit();
      }

      @Test (priority = 4, groups =  {"E2E", "Sanity", "Regression"})
      public void loginTest() {

            String freecrmURL = "https://ui.cogmento.com/";
            driver.get(freecrmURL);
            driver.manage().window().maximize();

            WebElement usernameElement =
driver.findElement(By.name("email"));
            WebElement passwordElement =
driver.findElement(By.name("password"));
            WebElement buttonElement =
driver.findElement(By.cssSelector("#ui > div > div > form > div >
div.ui.fluid.large.blue.submit.button"));

            usernameElement.sendKeys("emad.naser1@gmail.com");
            passwordElement.sendKeys("123456");
            buttonElement.click();

        WebDriverWait wait = new WebDriverWait(driver,
Duration.ofSeconds(10));
```

```
        WebElement userDisplayElement =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelecto
r(".user-display")));
           String ActualResultString =  userDisplayElement.getText();
           String ExpectedResultString = "Emad Nasser"; // Pass
           System.out.println(ActualResultString.equals(
                              ExpectedResultString));
           Assert.assertEquals(ActualResultString,
ExpectedResultString, "Username or password are not correct! ");

      }

      @BeforeMethod (groups =  {"Sanity","E2E","Regression"})
      public void setUP() {
           String ChromeDriverPath = "D:\\PC\\chromedriver-
win64\\chromedriver.exe";
           String chromeDriverKey = "webdriver.chrome.driver";
           System.setProperty(chromeDriverKey, ChromeDriverPath);
           driver = new ChromeDriver();
      }

      @AfterMethod (groups = {"Sanity","E2E","Regression"})
      public void tearDown() {
           driver.quit();
      }

}
```

Parameters in TestNG

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="hello test">
 <parameter name ="URL" value="https://ui.cogmento.com/"></parameter>
 <test thread-count="5" name="End to End Test">
  <groups>
   <run>
     <include name ="Sanity"></include>
   </run>
  </groups>
  <classes>
   <class name="com.freecrm.testcases.HomePage"></class>
   <class name="com.freecrm.testcases.LoginPageTest"></class>
  </classes>
 </test> <!-- End to End Test -->
 </suite> <!-- hello test -->
```

```
@Parameters ({"URL"})
      @BeforeMethod (groups =  {"Sanity","E2E","Regression"})
      public void setUP(String URL) {
            String ChromeDriverPath = "D:\\PC\\chromedriver-
win64\\chromedriver.exe";
            String chromeDriverKey = "webdriver.chrome.driver";
            System.setProperty(chromeDriverKey, ChromeDriverPath);
            driver = new ChromeDriver();
            driver.get(URL);
      }
```

```
@Test (priority = 1, groups = {"E2E"})
      public void titleTest() {

String ExpectedResultString = "Cogmento CRM";
String ActualResultString = driver.getTitle(); // "Cogmento CRM"

System.out.println(ExpectedResultString.equals(ActualResultString));
Assert.assertEquals(ActualResultString, ExpectedResultString);


      }
```

```
@Parameters ({"URL"})
@BeforeMethod (groups =  {"Sanity","E2E","Regression"})
public void login(String URL) {
String ChromeDriverPath = "D:\\PC\\chromedriver-
win64\\chromedriver.exe";
      String chromeDriverKey = "webdriver.chrome.driver";
      System.setProperty(chromeDriverKey, ChromeDriverPath);
      driver = new ChromeDriver();
      driver.get(URL);
      driver.manage().window().maximize();
      driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10))
;
      WebElement usernameElement =
driver.findElement(By.name("email"));
      WebElement passwordElement =
driver.findElement(By.name("password"));
      WebElement buttonElement = driver.findElement(By.cssSelector("#ui
> div > div > form > div > div.ui.fluid.large.blue.submit.button"));
      usernameElement.sendKeys("emad.naser1@gmail.com");
      passwordElement.sendKeys("123456");
      buttonElement.click();
      }
```

DataProvider in TestNG:

The **loginTest** method (the test case) will be executed 4 times (the number of data in DataProvider)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="hello test">
 <parameter name ="URL" value="https://ui.cogmento.com/"></parameter>
 <test thread-count="5" name="End to End Test">
  <groups>
   <run>
     <include name ="Sanity"></include>
   </run>
  </groups>
  <classes>
   <class name="com.freecrm.testcases.LoginPageTest"></class>
  </classes>
 </test> <!-- End to End Test -->
 </suite> <!-- hello test -->
```

	0	2
0	emad.naser1@gmail.com	Emad@123
1	emad.naser@gmail.com	Emad12345
2	emad.nas@gmail.com	Emad123
3	emad.naser@gmail.com	Ema123

```
@DataProvider
      public void MayData() {

            Object data [][] = new Object [4][2];

            data[0][0] = "emad.naser1@gmail.com";
            data[0][1] = "123456";
            data[1][0] = "emad.naser@gmail.com";
            data[1][1] = "Emad12345";
            data[2][0] = "emad.nas@gmail.com";
            data[3][0] = "emad.naser@gmail.com";
            data[3][1] = "Ema123";

      }
```

```
      @Test (priority = 4, groups =  {"E2E", "Sanity", "Regression"},
dataProvider = "MayData")
      public void loginTest(String username, String password) {

            driver.manage().window().maximize();

            WebElement usernameElement =
driver.findElement(By.name("email"));
            WebElement passwordElement =
driver.findElement(By.name("password"));
            WebElement buttonElement =
driver.findElement(By.cssSelector("#ui > div > div > form > div >
div.ui.fluid.large.blue.submit.button"));
            usernameElement.sendKeys(username); // from dataProvider
            passwordElement.sendKeys(password); // from dataProvider
            buttonElement.click();

         WebDriverWait wait = new WebDriverWait(driver,
Duration.ofSeconds(10));
```

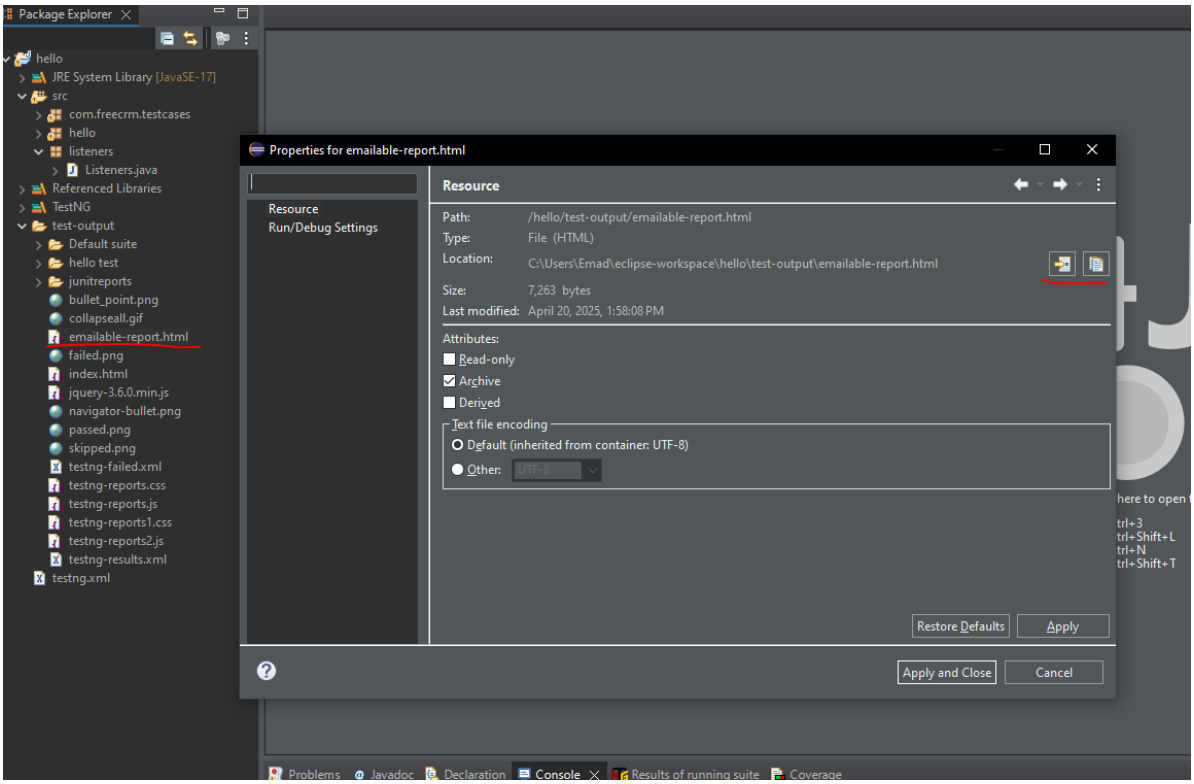
```
        WebElement userDisplayElement =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelecto
r(".user-display")));
           String ActualResultString =  userDisplayElement.getText();

           String ExpectedResultString = "Emad Nasser"; // Pass

     System.out.println(ActualResultString.equals(ExpectedResultString
));
           Assert.assertEquals(ActualResultString,
ExpectedResultString, "Username or password are not correct! ");


     }
```

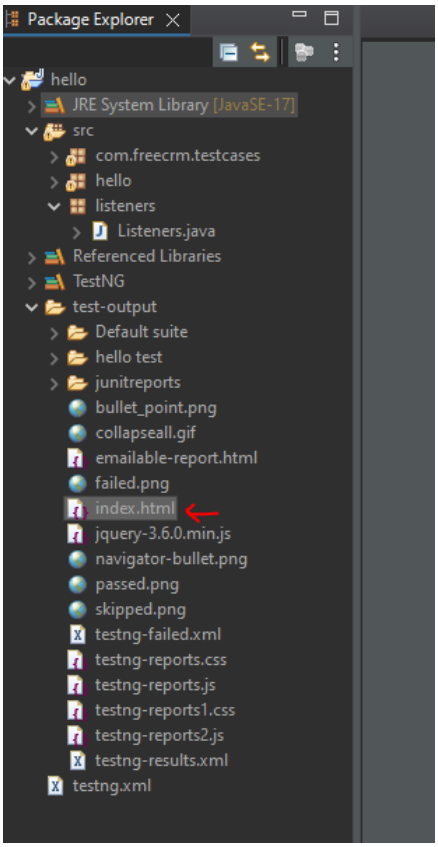
Listeners in TestNG

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="hello test">
  <listeners>
    <listener class-name="listeners.Listeners"/>
  </listeners>
 <parameter name ="URL" value="https://ui.cogmento.com/"></parameter>
 <test thread-count="5" name="End to End Test">
  <groups>
   <run>
     <include name ="Sanity"></include>
   </run>
  </groups>
  <classes>
   <class name="com.freecrm.testcases.LoginPageTest"></class>
  </classes>
 </test> <!-- End to End Test -->
 </suite> <!-- hello test -->
```

```java
package listeners;
import org.testng.ITestContext;
import org.testng.ITestListener;
import org.testng.ITestResult;

public class Listeners implements ITestListener {

      @Override
      public void onTestStart(ITestResult result) {
          System.out.println("Test has been started!");
        }

      @Override
        public void onTestSuccess(ITestResult result) {
            System.out.println("Test PASS!");
        }

      @Override
        public void onTestFailure(ITestResult result) {
            System.out.println("Test FAILED!");
        }


      @Override
        public void onTestSkipped(ITestResult result) {
            System.out.println("Test SKIPPED!");
        }


      @Override
        public void onStart(ITestContext context) {
            System.out.println("Test Start!");
        }

       @Override
        public void onFinish(ITestContext context) {
        System.out.println("Test Finish!");
        }

}
```

Reports in TestNG





Enabled annotation

The following method won't be executed because we add this: (enabled = false)

```
     @Test (priority = 2, enabled = false)
     public void titleTest1() {

           // the result of the following test is fail

           SoftAssert softAssert = new SoftAssert();

           String freecrmURL = "https://ui.cogmento.com/";
           driver.get(freecrmURL);

           String ExpectedResultString = "Cogmento";

           String ActualResultString = driver.getTitle(); // "Cogmento
CRM"


     System.out.println(ExpectedResultString.equals(ActualResultString
));
                // Output: true

           softAssert.assertEquals(ActualResultString,
ExpectedResultString); // Otput: fail
           //driver.quit();

           softAssert.assertAll(); // You must add assertAll(); in
order to print Pass  / Fail, otherwise will print pass even though the
test is fail.

           System.out.println("This line was printed eventhough the
output is false");

     }
```

Create Test Package and Class:

TestBase Class:

```
package com.freecrm.base;

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

public class TestBase {

	public static WebDriver driver;

	public void initialization(String URL) {
		String ChromeDriverPath = "D:\\PC\\chromedriver-
win64\\chromedriver.exe";
		String chromeDriverKey = "webdriver.chrome.driver";
		System.setProperty(chromeDriverKey, ChromeDriverPath);
		driver = new ChromeDriver();
		driver.get(URL);
	}
}
```

LoginPageTest Class:

```
package com.freecrm.testcases;
import java.time.Duration;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.Assert;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;
import com.freecrm.base.TestBase;

public class LoginPageTest extends TestBase {

	@Test (priority = 1)
	public void titleTest() {

		String ExpectedResultString = "Cogmento CRM";
		String ActualResultString = driver.getTitle(); // "Cogmento
CRM"
```

```
	System.out.println(ExpectedResultString.equals(ActualResultString
));
		Assert.assertEquals(ActualResultString,
ExpectedResultString);
	}

	@Test (priority = 2, enabled = false)
	public void titleTest1() {

		// the result of the following test is fail
		SoftAssert softAssert = new SoftAssert();
		String freecrmURL = "https://ui.cogmento.com/";
		driver.get(freecrmURL);
		String ExpectedResultString = "Cogmento";
		String ActualResultString = driver.getTitle(); // "Cogmento
CRM"

	System.out.println(ExpectedResultString.equals(ActualResultString
));
		softAssert.assertEquals(ActualResultString,
ExpectedResultString); // Otput: fail
		softAssert.assertAll(); // You must add assertAll(); in
order to print Pass  / Fail, otherwise will print pass even though the
test is fail.
		System.out.println("This line was printed eventhough the
output is false");

	}

	@Test (priority = 2)
	public void urlCheck() {

		//String freecrmURL = "https://ui.cogmento.com/";
		//driver.get(freecrmURL);
		String ExpectedResultString = "https://ui.cogmento.com/";
		String ActualResultString = driver.getCurrentUrl();


	System.out.println(ExpectedResultString.equals(ActualResultString
));
		Assert.assertEquals(ActualResultString,
ExpectedResultString);

		//driver.quit();
	}

	@Test (priority = 3)
```

```
     public void logoTest() {

          String freecrmURL =
"https://classic.freecrm.com/index.html";
          driver.get(freecrmURL);
          WebElement logoElement =
driver.findElement(By.xpath("//img[@src='https://classic.freecrm.com/i
mg/logo.png']"));
          boolean ExpectedResultBoolean = true;
          boolean ActualResultBoolean = logoElement.isDisplayed();
          System.out.println(ExpectedResultBoolean ==
ActualResultBoolean);
          Assert.assertEquals(ActualResultBoolean,
ExpectedResultBoolean);

          //driver.quit();
     }


     @Test (priority = 4)
     public void loginTest() {

          //String freecrmURL = "https://ui.cogmento.com/";
          //driver.get(freecrmURL);
          driver.manage().window().maximize();

          WebElement usernameElement =
driver.findElement(By.name("email"));
          WebElement passwordElement =
driver.findElement(By.name("password"));
          WebElement buttonElement =
driver.findElement(By.cssSelector("#ui > div > div > form > div >
div.ui.fluid.large.blue.submit.button"));
          String username = "emad.naser1@gmail.com";
          String password = "123456";
          usernameElement.sendKeys(username);
          passwordElement.sendKeys(password);
          buttonElement.click();

        WebDriverWait wait = new WebDriverWait(driver,
Duration.ofSeconds(10));

        WebElement userDisplayElement =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelecto
r(".user-display")));
          String ActualResultString =  userDisplayElement.getText();

          String ExpectedResultString = "Emad Nasser"; // Pass
```

```
            //String ExpectedResultString = "Emad "; // Output error:
expected [Emad ] but found [Emad Nasser]


      System.out.println(ActualResultString.equals(ExpectedResultString
));
            Assert.assertEquals(ActualResultString,
ExpectedResultString, "Username or password are not correct! ");
      }

      @DataProvider
      public Object [][] MayData() {

            Object data [][] = new Object [4][2];

            data[0][0] = "emad.naser1@gmail.com";
            data[0][1] = "123456";
            data[1][0] = "emad.naser@gmail.com";
            data[1][1] = "Emad12345";
            data[2][0] = "emad.nas@gmail.com";
            data[3][0] = "emad.naser@gmail.com";
            data[3][1] = "Ema123";

            return data;

      }

      @BeforeMethod
      public void setUP() {
            initialization("https://ui.cogmento.com");
      }

      @AfterMethod
      public void tearDown() {
            driver.quit();
      }

}
```

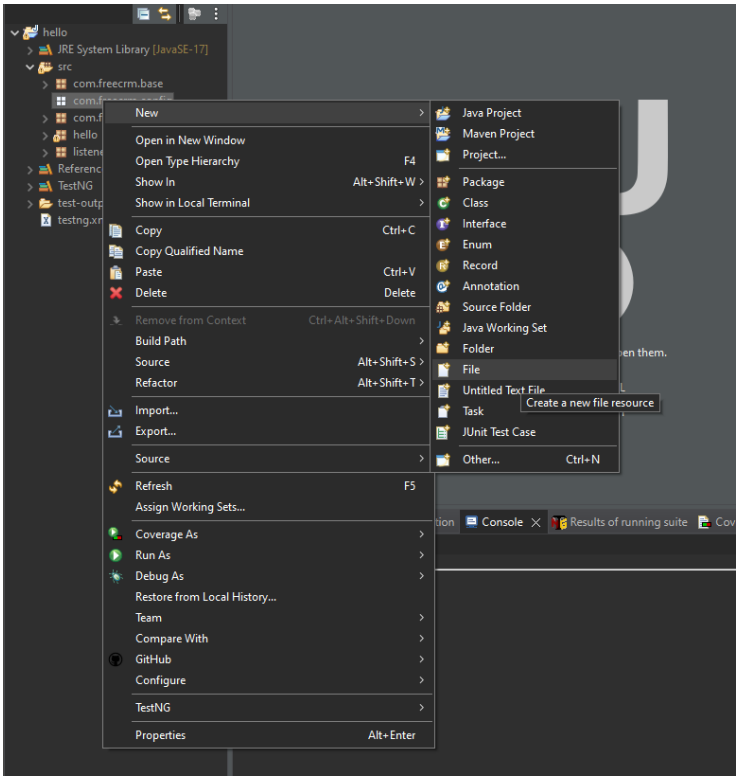
HomePage Class:

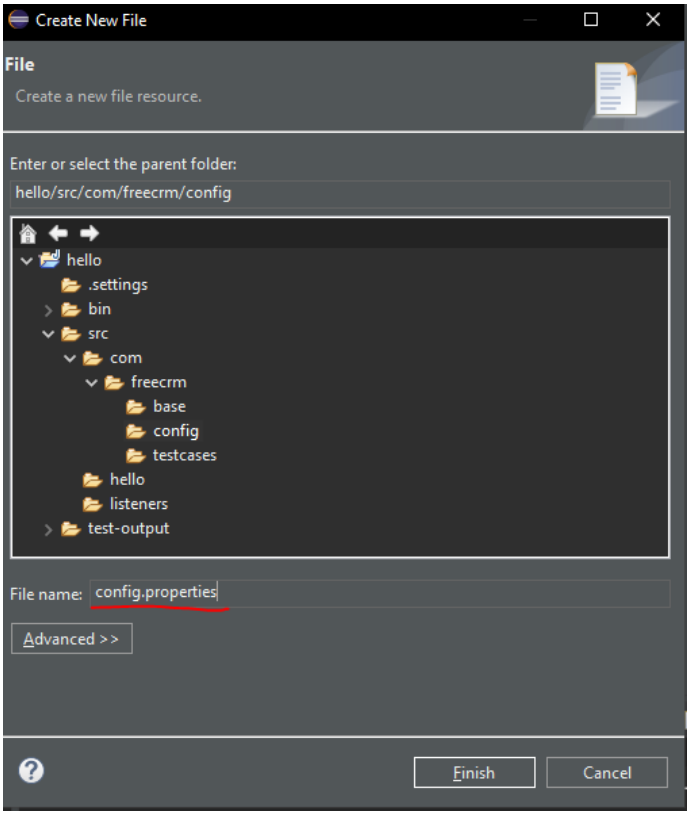
```
package com.freecrm.testcases;
import static org.testng.Assert.assertTrue;
import java.time.Duration;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;
import com.freecrm.base.TestBase;

public class HomePage extends TestBase {

	@BeforeMethod
	public void login( ) {
		initialization("https://ui.cogmento.com");
		driver.manage().window().maximize();

	driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10))
;
		WebElement usernameElement =
driver.findElement(By.name("email"));
		WebElement passwordElement =
driver.findElement(By.name("password"));
		WebElement buttonElement =
driver.findElement(By.cssSelector("#ui > div > div > form > div >
div.ui.fluid.large.blue.submit.button"));
		usernameElement.sendKeys("emad.naser1@gmail.com");
		passwordElement.sendKeys("123456");
		buttonElement.click();

	}

	@AfterMethod
	public void tearDown() {
		driver.quit();
	}

	@Test
	public void clickOnContactTest() {
		WebElement contactsElement =
driver.findElement(By.xpath("//span[text()='Contacts']"));
		contactsElement.click();
		WebElement statusLablElement =
driver.findElement(By.xpath("//th[text()='Status']"));
		boolean ActualResultBooloean =
statusLablElement.isDisplayed();
```

```
            assertTrue(ActualResultBooloean, "The status lable text not
displayed");

      }
      @Test
      public void clickOnDealsTest() {
            WebElement contactsElement =
driver.findElement(By.xpath("//span[text()='Deals']"));
            contactsElement.click();
            WebElement statusLablElement =
driver.findElement(By.xpath("//th[text()='Status']"));
            boolean ActualResultBooloean =
statusLablElement.isDisplayed();
            assertTrue(ActualResultBooloean, "The status lable text not
displayed");

      }

      @Test
      public void clickOnTasksTest() {
            WebElement contactsElement =
driver.findElement(By.xpath("//span[text()='Tasks']"));
            contactsElement.click();
            WebElement statusLablElement =
driver.findElement(By.xpath("//th[text()='Status']"));
            boolean ActualResultBooloean =
statusLablElement.isDisplayed();
            assertTrue(ActualResultBooloean, "The status lable text not
displayed");
      }

}
```

Create Properties File, global environment:

Package -> right click -> New -> File -> name with the extension .properties -> Finish





config.properties:

```
URL=https://ui.cogmento.com/
username=emad.naser1@gmail.com
password=123456
```

<u>HomePage class:</u>

```
package com.freecrm.testcases;
import static org.testng.Assert.assertTrue;

import java.io.IOException;
import java.time.Duration;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;
import com.freecrm.base.TestBase;

public class HomePage extends TestBase {

	public HomePage() throws IOException {
		super(); // call the constructor from the base class
		// TODO Auto-generated constructor stub
	}

	@BeforeMethod
	public void login( ) {
		initialization();

		WebElement usernameElement =
driver.findElement(By.name("email"));
		WebElement passwordElement =
driver.findElement(By.name("password"));
		WebElement buttonElement =
driver.findElement(By.cssSelector("#ui > div > div > form > div >
div.ui.fluid.large.blue.submit.button"));
		usernameElement.sendKeys("emad.naser1@gmail.com");
		passwordElement.sendKeys("123456");
		buttonElement.click();
	}

	@AfterMethod
	public void tearDown() {
		driver.quit();
	}
```

```
	@Test
	public void clickOnContactTest() {
		WebElement contactsElement =
driver.findElement(By.xpath("//span[text()='Contacts']"));
		contactsElement.click();
		WebElement statusLablElement =
driver.findElement(By.xpath("//th[text()='Status']"));
		boolean ActualResultBooloean =
statusLablElement.isDisplayed();
		assertTrue(ActualResultBooloean, "The status lable text not
displayed");

	}
	@Test
	public void clickOnDealsTest() {
		WebElement contactsElement =
driver.findElement(By.xpath("//span[text()='Deals']"));
		contactsElement.click();
		WebElement statusLablElement =
driver.findElement(By.xpath("//th[text()='Status']"));
		boolean ActualResultBooloean =
statusLablElement.isDisplayed();
		assertTrue(ActualResultBooloean, "The status lable text not
displayed");

	}

	@Test
	public void clickOnTasksTest() {
		WebElement contactsElement =
driver.findElement(By.xpath("//span[text()='Tasks']"));
		contactsElement.click();
		WebElement statusLablElement =
driver.findElement(By.xpath("//th[text()='Status']"));
		boolean ActualResultBooloean =
statusLablElement.isDisplayed();
		assertTrue(ActualResultBooloean, "The status lable text not
displayed");
	}

}
```

LoginPageTest Class:

```
package com.freecrm.testcases;
import java.io.IOException;
import java.time.Duration;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.Assert;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;
import com.freecrm.base.TestBase;

public class LoginPageTest extends TestBase {

	public LoginPageTest() throws IOException {
		super(); // call the constructor from the base class
		// TODO Auto-generated constructor stub
	}

	@Test (priority = 1)
	public void titleTest() {

		String ExpectedResultString = "Cogmento CRM";
		String ActualResultString = driver.getTitle(); // "Cogmento
CRM"

	System.out.println(ExpectedResultString.equals(ActualResultString
));
		Assert.assertEquals(ActualResultString,
ExpectedResultString);
	}


	@Test (priority = 2)
	public void urlCheck() {

		String ExpectedResultString = "https://ui.cogmento.com/";
		String ActualResultString = driver.getCurrentUrl();


	System.out.println(ExpectedResultString.equals(ActualResultString
));
```

```
            Assert.assertEquals(ActualResultString,
ExpectedResultString);

      }

      @Test (priority = 3)
      public void logoTest() {

            String freecrmURL =
"https://classic.freecrm.com/index.html";
            driver.get(freecrmURL);
            WebElement logoElement =
driver.findElement(By.xpath("//img[@src='https://classic.freecrm.com/i
mg/logo.png']"));
            boolean ExpectedResultBoolean = true;
            boolean ActualResultBoolean = logoElement.isDisplayed();
            System.out.println(ExpectedResultBoolean ==
ActualResultBoolean);
            Assert.assertEquals(ActualResultBoolean,
ExpectedResultBoolean);
      }


      @Test (priority = 4)
      public void loginTest() {

            WebElement usernameElement =
driver.findElement(By.name("email"));
            WebElement passwordElement =
driver.findElement(By.name("password"));
            WebElement buttonElement =
driver.findElement(By.cssSelector("#ui > div > div > form > div >
div.ui.fluid.large.blue.submit.button"));
            String username = "emad.naser1@gmail.com";
            String password = "123456";
            usernameElement.sendKeys(username);
            passwordElement.sendKeys(password);
            buttonElement.click();

        WebDriverWait wait = new WebDriverWait(driver,
Duration.ofSeconds(10));

        WebElement userDisplayElement =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelecto
r(".user-display")));
            String ActualResultString =  userDisplayElement.getText();

            String ExpectedResultString = "Emad Nasser";
```

```
	System.out.println(ActualResultString.equals(ExpectedResultString
));
		Assert.assertEquals(ActualResultString,
ExpectedResultString, "Username or password are not correct! ");
	}

	@BeforeMethod
	public void setUP() {
		initialization();
	}

	@AfterMethod
	public void tearDown() {
		driver.quit();
	}
}
```

Class TestBase:

```
package com.freecrm.base;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.time.Duration;
import java.util.Properties;

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

public class TestBase {

	public static WebDriver driver;

	public static Properties properties;

	// constructor
	public TestBase () throws IOException
	{
		properties = new Properties();

		try
		{
			FileInputStream file = new
FileInputStream("C:\\Users\\Emad\\eclipse-
workspace\\hello\\src\\com\\freecrm\\config\\config.properties");
			properties.load(file);
```

```
            }

            catch (FileNotFoundException e)
            {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
      }

      public void initialization() {

            String ChromeDriverPath = "D:\\PC\\chromedriver-
win64\\chromedriver.exe";
            String chromeDriverKey = "webdriver.chrome.driver";
            System.setProperty(chromeDriverKey, ChromeDriverPath);
            driver = new ChromeDriver();
            driver.get(properties.getProperty("URL"));
            driver.manage().window().maximize();

      driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10))
;

      }
}
```

Build WebDriver Event Listener

```
package com.freecrm.util;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.events.WebDriverListener;

public class WebListener implements WebDriverListener
{

      @Override
         public void beforeGet(WebDriver driver, String url) {
             System.out.println("Before navigating to: " + url);
         }

         @Override
         public void afterGet(WebDriver driver, String url) {
             System.out.println("Navigated to: " + url);
         }

         @Override
         public void beforeGetCurrentUrl(WebDriver driver) {
             System.out.println("Trying to get current URL");
         }

         @Override
         public void afterGetCurrentUrl(WebDriver driver, String
result) {
             System.out.println("Current URL is: " + result);
         }

         @Override
         public void beforeGetTitle(WebDriver driver) {
             System.out.println("Trying to get page title");
         }

         @Override
         public void afterGetTitle(WebDriver driver, String result) {
             System.out.println("Page title is: " + result);
         }

         @Override
         public void beforeFindElement(WebDriver driver, By locator) {
             System.out.println("Trying to find element by: " +
locator.toString());
```

```
        }

        @Override
        public void afterFindElement(WebDriver driver, By locator,
WebElement result) {
            System.out.println("Found element by: " +
locator.toString());
        }

}
```

```
package com.freecrm.base;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.time.Duration;
import java.util.Properties;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.events.EventFiringDecorator;

import com.freecrm.util.WebListener;

public class TestBase {

    public static WebDriver driver;

    public static Properties properties;

    // constructor
    public TestBase () throws IOException
    {
        properties = new Properties();

        try
        {
            FileInputStream file = new
FileInputStream("C:\\Users\\Emad\\eclipse-
workspace\\hello\\src\\com\\freecrm\\config\\config.properties");
            properties.load(file);
        }

        catch (FileNotFoundException e)
        {
            // TODO Auto-generated catch block
```

```
                e.printStackTrace();
            }
        }

        public void initialization() {

            String ChromeDriverPath = "D:\\PC\\chromedriver-
win64\\chromedriver.exe";
            String chromeDriverKey = "webdriver.chrome.driver";
            System.setProperty(chromeDriverKey, ChromeDriverPath);

            WebDriver baseDriver = new ChromeDriver();

            // Attach listener
            WebListener listener = new WebListener();
            driver = new
EventFiringDecorator(listener).decorate(baseDriver);

            // Regular setup
            driver.get(properties.getProperty("URL"));
            driver.manage().window().maximize();

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

        }
}
```

```
package com.freecrm.testcases;
import java.io.IOException;
import java.time.Duration;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.Assert;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;
import com.freecrm.base.TestBase;

public class LoginPageTest extends TestBase {

        public LoginPageTest() throws IOException {
            super(); // call the constructor from the base class
            // TODO Auto-generated constructor stub
```

```
      }

      @Test (priority = 1)
      public void titleTest() {

            String ExpectedResultString = "Cogmento CRM";
            String ActualResultString = driver.getTitle(); // "Cogmento
CRM"

      System.out.println(ExpectedResultString.equals(ActualResultString
));
            Assert.assertEquals(ActualResultString,
ExpectedResultString);
      }

      @Test (priority = 2)
      public void urlCheck() {

            String ExpectedResultString = "https://ui.cogmento.com/";
            String ActualResultString = driver.getCurrentUrl();


      System.out.println(ExpectedResultString.equals(ActualResultString
));
            Assert.assertEquals(ActualResultString,
ExpectedResultString);

      }

      @Test (priority = 3)
      public void logoTest() {

            String freecrmURL =
"https://classic.freecrm.com/index.html";
            driver.get(freecrmURL);
            WebElement logoElement =
driver.findElement(By.xpath("//img[@src='https://classic.freecrm.com/i
mg/logo.png']"));
            boolean ExpectedResultBoolean = true;
            boolean ActualResultBoolean = logoElement.isDisplayed();
            System.out.println(ExpectedResultBoolean ==
ActualResultBoolean);
            Assert.assertEquals(ActualResultBoolean,
ExpectedResultBoolean);
      }


      @Test (priority = 4)
```

```
      public void loginTest() {

            WebElement usernameElement =
driver.findElement(By.name("email"));
            WebElement passwordElement =
driver.findElement(By.name("password"));
            WebElement buttonElement =
driver.findElement(By.cssSelector("#ui > div > div > form > div >
div.ui.fluid.large.blue.submit.button"));
            String username = "emad.naser1@gmail.com";
            String password = "123456";
            usernameElement.sendKeys(username);
            passwordElement.sendKeys(password);
            buttonElement.click();

        WebDriverWait wait = new WebDriverWait(driver,
Duration.ofSeconds(10));

        WebElement userDisplayElement =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelecto
r(".user-display")));
            String ActualResultString =  userDisplayElement.getText();

            String ExpectedResultString = "Emad Nasser";

      System.out.println(ActualResultString.equals(ExpectedResultString
));
            Assert.assertEquals(ActualResultString,
ExpectedResultString, "Username or password are not correct! ");
      }

      @BeforeMethod
      public void setUP() {
            initialization();
      }

      @AfterMethod
      public void tearDown() {
            driver.quit();
      }

      @Test (priority = 2, enabled = false)
      public void titleTest1() {

            // the result of the following test is fail
            SoftAssert softAssert = new SoftAssert();
            String freecrmURL = "https://ui.cogmento.com/";
            driver.get(freecrmURL);
```

```
            String ExpectedResultString = "Cogmento";
            String ActualResultString = driver.getTitle(); // "Cogmento
CRM"

      System.out.println(ExpectedResultString.equals(ActualResultString
));
            softAssert.assertEquals(ActualResultString,
ExpectedResultString); // Otput: fail
            softAssert.assertAll(); // You must add assertAll(); in
order to print Pass  / Fail, otherwise will print pass even though the
test is fail.
            System.out.println("This line was printed eventhough the
output is false");

      }

      @DataProvider
      public Object [][] MayData() {

            Object data [][] = new Object [4][2];

            data[0][0] = "emad.naser1@gmail.com";
            data[0][1] = "123456";
            data[1][0] = "emad.naser@gmail.com";
            data[1][1] = "Emad12345";
            data[2][0] = "emad.nas@gmail.com";
            data[3][0] = "emad.naser@gmail.com";
            data[3][1] = "Ema123";

            return data;
      }

}
```

```
package com.freecrm.testcases;
import static org.testng.Assert.assertTrue;
import java.io.IOException;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;
import com.freecrm.base.TestBase;

public class HomePage extends TestBase {
```

```
	public HomePage() throws IOException {
		super(); // call the constructor from the base class
		// TODO Auto-generated constructor stub
	}

	@BeforeMethod
	public void login( ) {
		initialization();

		WebElement usernameElement =
driver.findElement(By.name("email"));
		WebElement passwordElement =
driver.findElement(By.name("password"));
		WebElement buttonElement =
driver.findElement(By.cssSelector("#ui > div > div > form > div >
div.ui.fluid.large.blue.submit.button"));
		usernameElement.sendKeys("emad.naser1@gmail.com");
		passwordElement.sendKeys("123456");
		buttonElement.click();
	}

	@AfterMethod
	public void tearDown() {
		driver.quit();
	}

	@Test
	public void clickOnContactTest() {
		WebElement contactsElement =
driver.findElement(By.xpath("//span[text()='Contacts']"));
		contactsElement.click();
		WebElement statusLablElement =
driver.findElement(By.xpath("//th[text()='Status']"));
		boolean ActualResultBooloean =
statusLablElement.isDisplayed();
		assertTrue(ActualResultBooloean, "The status lable text not
displayed");

	}
	@Test
	public void clickOnDealsTest() {
		WebElement contactsElement =
driver.findElement(By.xpath("//span[text()='Deals']"));
		contactsElement.click();
		WebElement statusLablElement =
driver.findElement(By.xpath("//th[text()='Status']"));
		boolean ActualResultBooloean =
statusLablElement.isDisplayed();
```

```
            assertTrue(ActualResultBooloean, "The status lable text not
displayed");

      }

      @Test
      public void clickOnTasksTest() {
            WebElement contactsElement =
driver.findElement(By.xpath("//span[text()='Tasks']"));
            contactsElement.click();
            WebElement statusLablElement =
driver.findElement(By.xpath("//th[text()='Status']"));
            boolean ActualResultBooloean =
statusLablElement.isDisplayed();
            assertTrue(ActualResultBooloean, "The status lable text not
displayed");
      }

}
```

Take Snapshot in Selenium:

The following lines are the standard code:

```
File srcFile =
((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(srcFile, new File("C:\\Users\\Emad\\eclipse-
workspace\\hello\\Snapshots\\titleTest.png"));
```

EX:

```
@Test (priority = 1)
      public void titleTest() throws IOException {

            String ExpectedResultString = "Cogmento CRM";
            String ActualResultString = driver.getTitle(); // "Cogmento
CRM"
      System.out.println(ExpectedResultString.equals(ActualResultString
));
Assert.assertEquals(ActualResultString, ExpectedResultString);

File srcFile =
((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
            FileUtils.copyFile(srcFile, new
File("C:\\Users\\Emad\\eclipse-
workspace\\hello\\Snapshots\\titleTest.png"));

      }
      @Test (priority = 2)
      public void urlCheck() throws IOException {

String ExpectedResultString = "https://ui.cogmento.com/";
String ActualResultString = driver.getCurrentUrl();

System.out.println(ExpectedResultString.equals(ActualResultString));
Assert.assertEquals(ActualResultString, ExpectedResultString);

File srcFile =
((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
            FileUtils.copyFile(srcFile, new
File("C:\\Users\\Emad\\eclipse-
workspace\\hello\\Snapshots\\urlCheck.png"));

      }
```

Now we will create a class and implement a method to get the name of every method to save the picture with that name:

```
public class TestUtils extends TestBase {

	public TestUtils() throws IOException {
		super();
		// TODO Auto-generated constructor stub
	}

	public static void takePicture(String name) throws IOException {
		File srcFile =
((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
		FileUtils.copyFile(srcFile, new
File("C:\\Users\\Emad\\eclipse-workspace\\hello\\Snapshots\\" + name +
".png"));

	}
}
```

```
	@Test (priority = 1)
	public void titleTest(Method method) throws IOException {

		String ExpectedResultString = "Cogmento CRM";
		String ActualResultString = driver.getTitle(); // "Cogmento
CRM"

	System.out.println(ExpectedResultString.equals(ActualResultString
));
		Assert.assertEquals(ActualResultString,
ExpectedResultString);
		TestUtils.takePicture(method.getName());

	}
```

The method now will get the name of the method (Test Case) and then pass it to the method takePicture and assign it to the parameter name.

Now we can add this line to all methods (Test Cases).

```
TestUtils.takePicture(method.getName());
```

Take Video for Selenium Test:

1. Using ATU Reporter Library (Java)

- ATU Reporter can capture screen recordings during Selenium tests.

2. Using Monte Screen Recorder (Java)

- Lightweight library that records screen or a browser window.
- You can start and stop recording in your test code:

ATU Test Recorder vs. Monte Recorder

	ATU Test Recorder	**Monte Recorder**
Setup	Very easy	Medium (more manual code)
Audio Recording	Optional	Possible but more setup
Custom Settings	Basic (just start/stop)	Full control
Recommended for	Quick Selenium Test Recording	Advanced/Custom recording

```
import atu.testrecorder.ATUTestRecorder;
import atu.testrecorder.exceptions.ATUTestRecorderException;

public class VideoExample {

     @Test
public void recordVideo() throws ATUTestRecorderException {

ATUTestRecorder recorder = new
ATUTestRecorder("C:\\Users\\Emad\\eclipse-
workspace\\hello\\Videos","testcase1",false);
           recorder.start();

String ChromeDriverPath = "D:\\PC\\chromedriver-
win64\\chromedriver.exe";
         String chromeDriverKey = "webdriver.chrome.driver";
        System.setProperty(chromeDriverKey, ChromeDriverPath);
        WebDriver driver = new ChromeDriver();
        driver.get("https://ui.cogmento.com/");
        driver.quit();
        recorder.stop();
     }
}
```

Update the project:

Class TestBase:

```
public class TestBase {

     public static WebDriver driver;
     public static Properties properties;
     public static ATUTestRecorder recorder;
...
}
```

Class LoginPageTest

```
package com.freecrm.testcases;
import java.io.IOException;
import java.lang.reflect.Method;
import java.time.Duration;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testng.Assert;
```

```java
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;
import com.freecrm.base.TestBase;
import com.freecrm.util.TestUtils;

import atu.testrecorder.ATUTestRecorder;
import atu.testrecorder.exceptions.ATUTestRecorderException;

public class LoginPageTest extends TestBase {

	public LoginPageTest() throws IOException {
		super(); // call the constructor from the base class
		// TODO Auto-generated constructor stub
	}

	@Test (priority = 1)
	public void titleTest(Method method) throws IOException {

		String ExpectedResultString = "Cogmento CRM";
		String ActualResultString = driver.getTitle(); // "Cogmento
CRM"

	System.out.println(ExpectedResultString.equals(ActualResultString
));
		Assert.assertEquals(ActualResultString,
ExpectedResultString);
		//TestUtils.takePicture(method.getName());

	}

	@Test (priority = 2)
	public void urlCheck(Method method) throws IOException {

		String ExpectedResultString = "https://ui.cogmento.com/";
		String ActualResultString = driver.getCurrentUrl();


	System.out.println(ExpectedResultString.equals(ActualResultString
));
		Assert.assertEquals(ActualResultString,
ExpectedResultString);
		//TestUtils.takePicture(method.getName());
	}

	@Test (priority = 3)
```

```
      public void logoTest(Method method) throws IOException {

            String freecrmURL =
"https://classic.freecrm.com/index.html";
            driver.get(freecrmURL);
            WebElement logoElement =
driver.findElement(By.xpath("//img[@src='https://classic.freecrm.com/i
mg/logo.png']"));
            boolean ExpectedResultBoolean = true;
            boolean ActualResultBoolean = logoElement.isDisplayed();
            System.out.println(ExpectedResultBoolean ==
ActualResultBoolean);
            Assert.assertEquals(ActualResultBoolean,
ExpectedResultBoolean);
            //TestUtils.takePicture(method.getName());
      }


      @Test (priority = 4)
      public void loginTest(Method method) throws IOException {

            WebElement usernameElement =
driver.findElement(By.name("email"));
            WebElement passwordElement =
driver.findElement(By.name("password"));
            WebElement buttonElement =
driver.findElement(By.cssSelector("#ui > div > div > form > div >
div.ui.fluid.large.blue.submit.button"));
            String username = "emad.naser1@gmail.com";
            String password = "123456";
            usernameElement.sendKeys(username);
            passwordElement.sendKeys(password);
            buttonElement.click();

        WebDriverWait wait = new WebDriverWait(driver,
Duration.ofSeconds(10));

        WebElement userDisplayElement =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelecto
r(".user-display")));
            String ActualResultString =  userDisplayElement.getText();

            String ExpectedResultString = "Emad Nasser";

      System.out.println(ActualResultString.equals(ExpectedResultString
));
            Assert.assertEquals(ActualResultString,
ExpectedResultString, "Username or password are not correct! ");
```

```
            //TestUtils.takePicture(method.getName());
      }

      @Test (priority = 2, enabled = false)
      public void titleTest1() {

            // the result of the following test is fail
            SoftAssert softAssert = new SoftAssert();
            String freecrmURL = "https://ui.cogmento.com/";
            driver.get(freecrmURL);
            String ExpectedResultString = "Cogmento";
            String ActualResultString = driver.getTitle(); // "Cogmento
CRM"

      System.out.println(ExpectedResultString.equals(ActualResultString
));
            softAssert.assertEquals(ActualResultString,
ExpectedResultString); // Otput: fail
            softAssert.assertAll(); // You must add assertAll(); in
order to print Pass  / Fail, otherwise will print pass even though the
test is fail.
            System.out.println("This line was printed eventhough the
output is false");

      }

      @DataProvider
      public Object [][] MayData() {

            Object data [][] = new Object [4][2];

            data[0][0] = "emad.naser1@gmail.com";
            data[0][1] = "123456";
            data[1][0] = "emad.naser@gmail.com";
            data[1][1] = "Emad12345";
            data[2][0] = "emad.nas@gmail.com";
            data[3][0] = "emad.naser@gmail.com";
            data[3][1] = "Ema123";

            return data;
      }

      @BeforeMethod
      public void setUP(Method method) throws ATUTestRecorderException
{
            initialization();
            recorder = new ATUTestRecorder("C:\\Users\\Emad\\eclipse-
workspace\\hello\\Videos", method.getName(),false);
```

```
            recorder.start();
      }

      @AfterMethod
      public void tearDown(Method method) throws
ATUTestRecorderException, IOException {
            TestUtils.takePicture(method.getName());
            driver.quit();
            recorder.stop();
      }
}
```

Class HomePage

```
package com.freecrm.testcases;
import static org.testng.Assert.assertTrue;
import java.io.IOException;
import java.lang.reflect.Method;

import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;
import com.freecrm.base.TestBase;
import com.freecrm.util.TestUtils;

import atu.testrecorder.ATUTestRecorder;
import atu.testrecorder.exceptions.ATUTestRecorderException;

public class HomePage extends TestBase {

      public HomePage() throws IOException {
            super(); // call the constructor from the base class
            // TODO Auto-generated constructor stub
      }

      @BeforeMethod
      public void login(Method method) throws IOException,
ATUTestRecorderException {
            initialization();
            recorder = new ATUTestRecorder("C:\\Users\\Emad\\eclipse-
workspace\\hello\\Videos", method.getName(),false);
            WebElement usernameElement =
driver.findElement(By.name("email"));
            WebElement passwordElement =
driver.findElement(By.name("password"));
```

```
            WebElement buttonElement =
driver.findElement(By.cssSelector("#ui > div > div > form > div >
div.ui.fluid.large.blue.submit.button"));
            usernameElement.sendKeys("emad.naser1@gmail.com");
            passwordElement.sendKeys("123456");
            buttonElement.click();
            //TestUtils.takePicture(method.getName());
      }

      @AfterMethod
      public void tearDown(Method method) throws
ATUTestRecorderException, IOException {
            TestUtils.takePicture(method.getName());
            driver.quit();
            recorder.stop();
      }


      @Test
      public void clickOnContactTest(Method method) throws IOException
{
            WebElement contactsElement =
driver.findElement(By.xpath("//span[text()='Contacts']"));
            contactsElement.click();
            WebElement statusLablElement =
driver.findElement(By.xpath("//th[text()='Status']"));
            boolean ActualResultBooloean =
statusLablElement.isDisplayed();
            assertTrue(ActualResultBooloean, "The status lable text not
displayed");
            //TestUtils.takePicture(method.getName());
      }

      @Test
      public void clickOnDealsTest(Method method) throws IOException {
            WebElement contactsElement =
driver.findElement(By.xpath("//span[text()='Deals']"));
            contactsElement.click();
            WebElement statusLablElement =
driver.findElement(By.xpath("//th[text()='Status']"));
            boolean ActualResultBooloean =
statusLablElement.isDisplayed();
            assertTrue(ActualResultBooloean, "The status lable text not
displayed");
            //TestUtils.takePicture(method.getName());

      }
```

```
	@Test
	public void clickOnTasksTest(Method method) throws IOException {
		WebElement contactsElement =
driver.findElement(By.xpath("//span[text()='Tasks']"));
		contactsElement.click();
		WebElement statusLablElement =
driver.findElement(By.xpath("//th[text()='Status']"));
		boolean ActualResultBooloean =
statusLablElement.isDisplayed();
		assertTrue(ActualResultBooloean, "The status lable text not
displayed");
		//TestUtils.takePicture(method.getName());
	}

}
```

Run test in multi-Browser:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="hello test">
 <test thread-count="5" name="End to End Test">
      <parameter name="browser" value="chrome"></parameter>
    <classes>
   <class name="com.freecrm.testcases.LoginPageTest"></class>
   <class name="com.freecrm.testcases.HomePage"></class>
  </classes>
 </test> <!-- End to End Test -->
 </suite> <!-- hello test -->
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="hello test">
 <test thread-count="5" name="End to End Test">
      <parameter name="browser" value="edge"></parameter>
    <classes>
   <class name="com.freecrm.testcases.LoginPageTest"></class>
   <class name="com.freecrm.testcases.HomePage"></class>
  </classes>
 </test> <!-- End to End Test -->
 </suite> <!-- hello test -->
```

Base Class:

```
      public void initialization(String browser) {
            String ChromeDriverPath = "D:\\PC\\chromedriver-
win64\\chromedriver.exe";
          String chromeDriverKey = "webdriver.chrome.driver";
          String EdgeDriverPath =
"D:\\PC\\edgedriver_win64\\msedgedriver.exe";
          String EdgeDriverKey = "webdriver.edge.driver";
          WebDriver baseDriver = null;

            if (browser.equalsIgnoreCase("chrome")) {
                  System.setProperty(chromeDriverKey,
ChromeDriverPath);
                  baseDriver  = new ChromeDriver();
            }

            else if (browser.equals("edge")) {
                  System.setProperty(EdgeDriverKey,EdgeDriverPath);
                  baseDriver = new EdgeDriver();
            }

        // Attach listener
        WebListener listener = new WebListener();
        driver = new
EventFiringDecorator(listener).decorate(baseDriver);

        // Regular setup
        driver.get(properties.getProperty("URL"));
        driver.manage().window().maximize();

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

      }
```

Login Class:

```
@Parameters({"browser"})
      @BeforeMethod
      public void setUP(String browser, Method method) throws
ATUTestRecorderException {
            initialization(browser);
            recorder = new ATUTestRecorder("C:\\Users\\Emad\\eclipse-
workspace\\hello\\Videos", method.getName(),false);
            recorder.start();
      }
```

HomePage Class:

```
@Parameters({"browser"})
      @BeforeMethod
      public void login(String browser, Method method) throws
IOException, ATUTestRecorderException {
            initialization(browser);
            recorder = new ATUTestRecorder("C:\\Users\\Emad\\eclipse-
workspace\\hello\\Videos", method.getName(),false);
            WebElement usernameElement =
driver.findElement(By.name("email"));
            WebElement passwordElement =
driver.findElement(By.name("password"));
            WebElement buttonElement =
driver.findElement(By.cssSelector("#ui > div > div > form > div >
div.ui.fluid.large.blue.submit.button"));
            usernameElement.sendKeys("emad.naser1@gmail.com");
            passwordElement.sendKeys("123456");
            buttonElement.click();
            //TestUtils.takePicture(method.getName());
      }
```

Read data from Excel Sheet:

Add Jar Files

- poi-5.2.3.jar
- poi-ooxml-5.2.3.jar
- poi-ooxml-schemas-5.2.3.jar
- xmlbeans-5.1.1.jar
- commons-collections4-4.4.jar
- commons-compress-1.21.jar
- **commons-io-2.11.0.jar** ⬅️ **VERY IMPORTANT!**
- log4j-api-2.17.1.jar
- log4j-core-2.17.1.jar
- curvesapi-1.06.jar

Ex:

```
package com.freecrm.testcases;
import java.io.File;
import java.io.FileInputStream;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class ExplainReadDataFromExcel {

      public static void main(String[] args) throws Exception {
                                // Correct the file path (no extra quotes inside)
        File file = new File("D:\\TestData.xlsx");

        // Read the file
        FileInputStream fis = new FileInputStream(file);

        // Create a Workbook instance
        XSSFWorkbook workbook = new XSSFWorkbook(fis);

        // Access the sheet named "Data"
        XSSFSheet sheet = workbook.getSheet("Data");

        // Read the value of the first cell (row 0, column 0)
        String cellValue = sheet.getRow(1).getCell(0).toString();

        // Print the value
        System.out.println("Cell Value: " + cellValue);


        int rows = sheet.getLastRowNum();
```

```java
        int col = sheet.getRow(0).getLastCellNum();

        Object data[][] = new Object[rows][col];

        for (int i = 0; i < rows; i++) {
                for (int j = 0; j < col; j++) {
                      data[i][j] = sheet.getRow(i).getCell(j);
                }
           }

        System.out.println(data[1][1]); // Nasser

        // Always close the workbook
        workbook.close();
      }
}
```