Why Automation Testing??

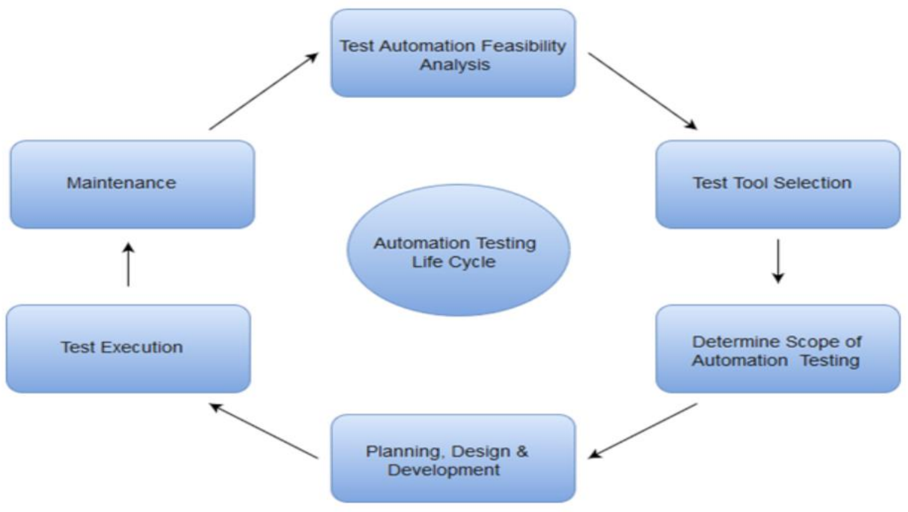
- It is required when we have a huge amount of regression test cases.
- It is required to save time & money
- It is required to increase the test coverage
- It is required to run tests anywhere & anytime
- It is required to generate robust reports
- It is required when we run tests with multiple sets of data
- It is required when testing manually is impossible
- It is required to test on several different hardware or software platforms and configurations

Selenium

Selenium is one of the most widely used open source and portable Web UI (User Interface) automation testing suite. It was originally developed by **Jason Huggins in 2004** as an internal tool at Thought Works.

- Selenium supports automation across different browsers, platforms and programming languages.
- Selenium can be easily deployed on platforms such as Windows, Linux, Solaris and Macintosh. Moreover, it supports OS (Operating System) for mobile applications like iOS, windows mobile and android.
- Selenium supports a variety of programming languages - include C#, Java, Perl, PHP, Python and Ruby.
- Selenium can be used to automate functional tests and can be integrated with automation test tools such as **Maven** & **Jenkins** to achieve continuous testing. It can also be integrated with tools such as **TestNG**, & **JUnit** for managing test cases and generating reports.

- It also supports parallel test execution which reduces time and increases the efficiency of tests.
- Selenium web driver does not require server installation, test scripts interact directly with the browser.


Test Automation Feasibility Analysis
Test Tool Selection
Determine Scope of Automation Testing
Planning, Design & Development
Test Execution
Maintenance
Automation Testing Life Cycle

Limitations of Selenium:

- Selenium does not support automation testing for desktop applications.
- Since Selenium is open source software, you have to rely on community forums to get your technical issues resolved.
- We can't perform automation tests on web services like SOAP or REST using Selenium.
- We should know at least one of the supported programming languages to create tests scripts in Selenium WebDriver.
- Selenium does not have any inbuilt reporting feature. It depends on the frameworks like **JUnit** and **TestNG** for test reports.
- It is not possible to perform testing on images. We need to integrate Selenium with **Sikuli** for image-based testing.
- Does not support Captcha, Barcodes.

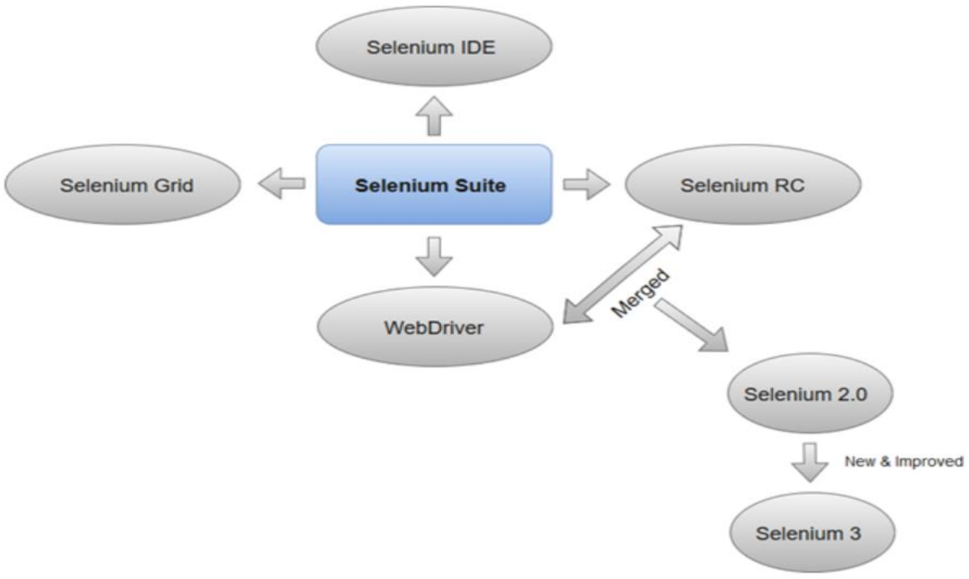
QTP (Quick Test Professional) / **UFT** (Unified Functional Testing) – By **Mercury Interactives**. Later acquired by HP.

Selenium Tool Suite

It is a suite of software, each with a different approach to support automation testing.

It includes Four major components:

1. Selenium Integrated Development Environment (**IDE**)
2. Selenium Remote Control (**RC**)
3. Selenium WebDriver
4. Selenium Grid



Selenium IDE (Integrated Development Environment)

- Selenium IDE is implemented as Firefox extension which provides record and playback functionality on test scripts.
- It allows testers to export recorded scripts in many languages like HTML, Java, Ruby, Python, C#, JUnit and TestNG.
- Selenium IDE has limited scope and the generated test scripts are not very robust and portable.

Selenium Remote Control (Selenium 1)

- Selenium RC (officially deprecated by selenium) allows testers to write automated web application UI test in any of the supported programming languages.
- It relies on JavaScript for automation. It supports Java, Javascript, Ruby, PHP, Python, Perl and C#. It supports almost every browser.
- It also involves an HTTP proxy server (Selenium RC Server & Selenium RC Client) which enables the browser to believe that the web application being tested comes from the domain provided by proxy server.

Selenium Grid

Selenium Grid allows us to run our tests simultaneously on different machines running different browsers and operating systems

- Selenium Grid follows the **Hub-Node Architecture** to achieve parallel execution of test scripts. The Hub is considered as master of the network and the other will be the nodes. Hub controls the execution of test scripts on various nodes of the network.

Selenium WebDriver

- Selenium WebDriver (Selenium 2) is the successor to Selenium RC and it is the most important component of Selenium Suite.
- Selenium WebDriver provides a programming interface to create and execute test cases.
- Test scripts are written in order to identify web elements on web pages and then desired actions are performed on those elements.
- Selenium WebDriver performs much faster as compared to Selenium RC because it makes direct calls to the web browsers. RC on the other hand needs an RC server to interact with the browser.
- We have separate drivers for each browser:
  - Mozilla Firefox Driver (Gecko Driver)
  - Google Chrome Driver
  - Internet Explorer Driver
  - Opera Driver
  - Safari Driver

- Selenium uses drivers, specific to each browser in order to establish a secure connection with the browser without revealing the internal logic of browser's functionality.
- When we execute a test script using WebDriver, the following operations are performed internally.
  - HTTP request is generated and sent to the browser driver for each Selenium command.
  - The driver receives the HTTP request through HTTP server.
  - HTTP Server decides all the steps to perform instructions which are executed on browser
  - Execution status is sent back to HTTP Server which is subsequently sent back to automation script.

Selenium Features

- Multiple Browser Support.
- Multiple Languages Support.
- Speed.
- Simple Commands.

Installation:

1. **Download and Install Java from oracle official site**

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Once the Java is installed, we need to set path or configure the environment variables in our system.

How to set environment variables:

Go to MyComputer properties >> Click on the advanced tab >> Click on environment variables >> Click on the new tab of System variables >> Write the 'path' in the 'variable name' >> Paste path of bin folder in the 'variable value' >> Click on ok button

Example : set path= C:\Program Files\Java\jdk-17\bin

2. **Download and Configure Eclipse IDE** – from the eclipse official site

https://www.eclipse.org/downloads/

and install "Eclipse IDE for Java Developers".

3. **Download Selenium WebDriver Java Client** – download it from Selenium official website. https://docs.seleniumhq.org/download/

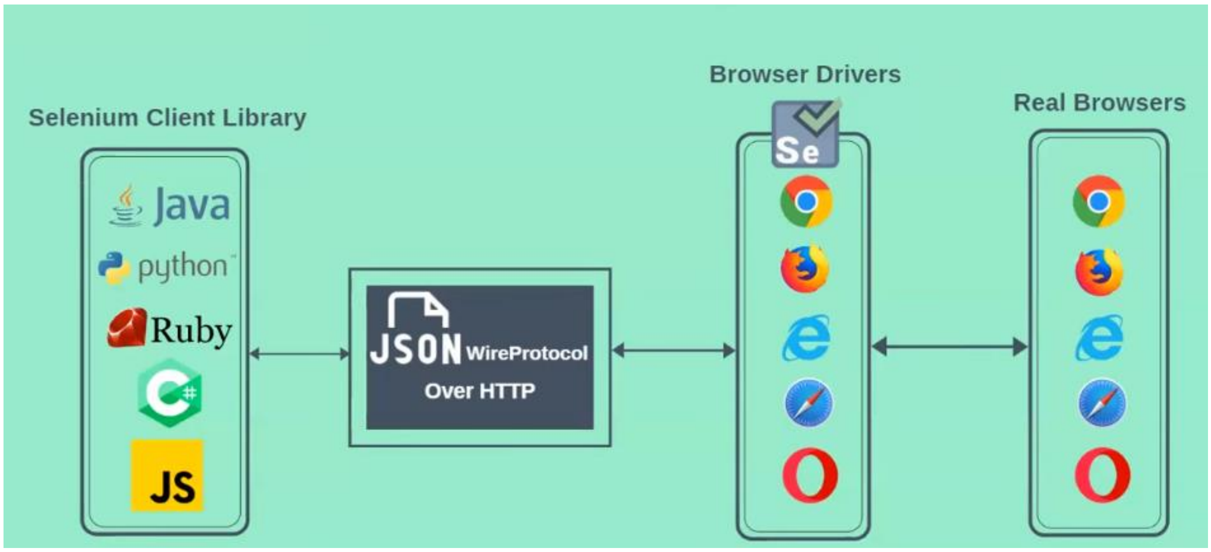
Select "Selenium Clients and WebDriver Language Bindings" for Java and download it. >>
The downloaded file would be in zipped format. Unzip the content in a folder. >> It contains the essential jar files required to configure Selenium WebDriver in Eclipse IDE.

4. **Configure Selenium WebDriver**

Launch Eclipse >> Create a new Java Project "SeleniumProject" >> Right click on the project SeleniumProject >> go to properties >> Click on "Java Build Path" >> Switch to Libraries tab and click on "Add External JARs" button >> Locate the directory where you have saved the selenium jars, select the respective jars and click on "Open" button >> Repeat the same steps for the jars which are present under the "libs" folder >> click on Apply and Close button.

With this, we have successfully configured Selenium WebDriver with Eclipse IDE. Now, we are ready to write our first test script and run in Eclipse IDE.

Selenium Architecture:



Create Object of Chrome, Firefox, Edge browsers.

WebDriver driver = new FirefoxDriver(); (Firefox browser)

WebDriver driver = new ChromeDriver(); (Chrome browser)

WebDriver driver = new EdgeDriver(); (Microsoft Edge browser)

Interview Que: Is WebDriver a class or interface.

Answer: WebDriver is an Interface.

ChromeDriver, FirefoxDriver, EdgeDriver, SafariDriver, RemoteWebDriver are the classes which are implementing WebDriver interface.

Interview Que: What is the super interface of WebDriver.

Answer: **SearchContext** is the super interface of the WebDriver interface.

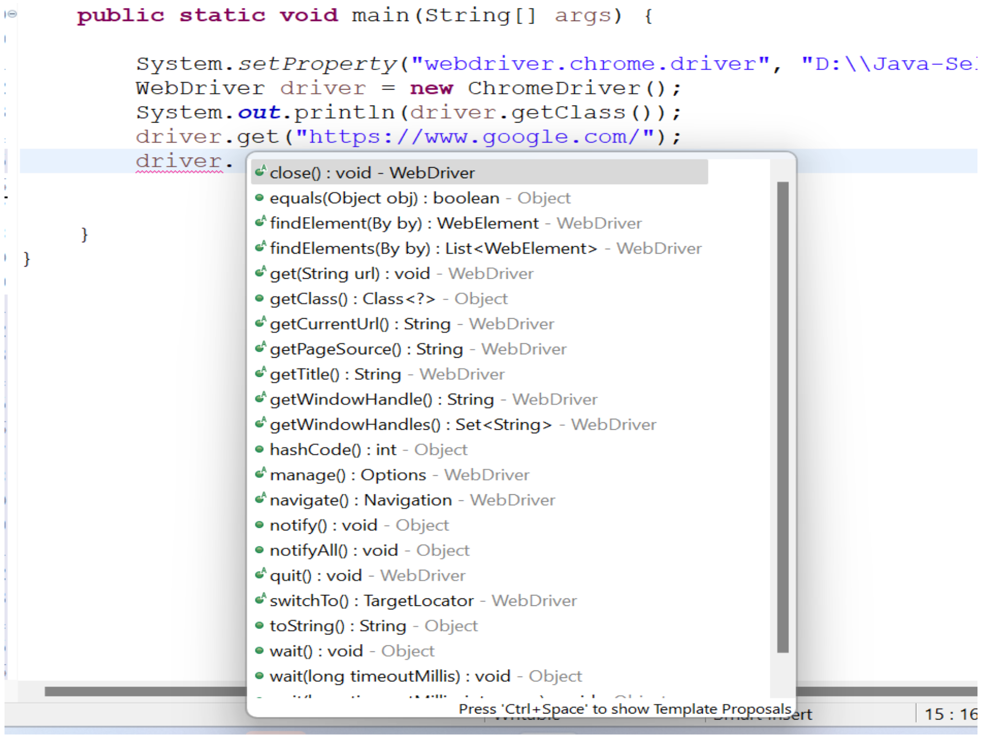
```
public abstract interface org.openqa.selenium.WebDriver
extends org.openqa.selenium.SearchContext
```

Selenium WebDriver- First Test Case

1. Launch Google Chrome browser.
2. Open URL: www.google.com
3. Click on the Google Search text box.
4. Type the value "HP Laptops"
5. Click on the Search button.

WebDriver Browser Commands:

Create a driver object for WebDriver and press the dot key. It will show you all of the possible methods provided by WebDriver.



```
    public static void main(String[] args) {

        System.setProperty("webdriver.chrome.driver", "D:\\Java-Se
        WebDriver driver = new ChromeDriver();
        System.out.println(driver.getClass());
        driver.get("https://www.google.com/");
        driver.

    }
}
```

```
close() : void - WebDriver
equals(Object obj) : boolean - Object
findElement(By by) : WebElement - WebDriver
findElements(By by) : List<WebElement> - WebDriver
get(String url) : void - WebDriver
getClass() : Class<?> - Object
getCurrentUrl() : String - WebDriver
getPageSource() : String - WebDriver
getTitle() : String - WebDriver
getWindowHandle() : String - WebDriver
getWindowHandles() : Set<String> - WebDriver
hashCode() : int - Object
manage() : Options - WebDriver
navigate() : Navigation - WebDriver
notify() : void - Object
notifyAll() : void - Object
quit() : void - WebDriver
switchTo() : TargetLocator - WebDriver
toString() : String - Object
wait() : void - Object
wait(long timeoutMillis) : void - Object
Press 'Ctrl+Space' to show Template Proposals
```

get(String args0) - This method loads a new web page in the existing browser window. It accepts String as parameter.

getTitle() - This method fetches the title of the current web page.

String Title = driver.getTitle();

getCurrentUrl() - This method fetches the Current URL of the current web page.

String CurrentUrl = driver.getCurrentUrl();

getPageSource() - This method returns the source code of the current web page loaded on the browser.

String PageSource = driver.getPageSource();

close() – This method closes only the current browser window.

quit() - This method closes all windows that WebDriver has opened.

Webdriver Navigation Commands:

navigate().to("url") – this method loads a new web page in the existing browser window.

[driver.get() never stores history and every time we use this command, the page will be refreshed. whereas driver.navigate().to() stores browser history so as to be used for other commands forward and back etc.]

navigate().forward() - This method enables the web browser to click on the **forward** button in the existing browser window.

navigate().back() – This method enables the web browser to click on the **back** button in the existing browser window.

navigate().refresh() - This method refresh/reloads the current web page.

Webdriver – WebElement commands: Web element refers to a HTML element

To get the web element object we have to write the below statement:

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

Here "APjFqb" is the value of id attribute.

clear() – This method is used to clear the user inputs from the text box.

sendKeys("text") – This method is used to set the user inputs.

click() – This method is used to perform click operation on any web element.

isDisplayed() – This method is used to check if an element is currently being displayed or not on the web page. This returns boolean value(true/false).

isEnabled() - This method determines if the element currently is Enabled or not. This returns boolean value(true/false).

isSelected() – Determine whether the element is selected or not. his operation only applies to input elements such as Checkboxes, Select Options and Radio Buttons.

submit() – This method works well / better than the click() if the current element is a form, or an element within a form.

getText() – This method will fetch the visible inner Text of the element. This will returns a String value.

getTagName() – This method gets the tag name of the Web Element.
getCssvalue() – This method Fetch CSS property value of the give element.

getAttribute(String Name) **-** This method gets the value of the given attribute of the element.

Locators:

Question: How to locate elements on the web page?

- Selenium uses element locators which helps in locating the web elements on the web page.
- Locators are the HTML properties of a web element which tells selenium about the web element it needs to perform the actions on.
- There is a diverse range of web elements like the textbox, Id, radio button etc. So selenium uses the locators to interact with the web elements on the web page.

Types of Locators:

1. Id
2. Name
3. Class name
4. Link text
5. Partial Link text
6. CSS selector
7. Tag name
8. Xpath

Locating web elements in WebDriver is performed with the help of **findElement()** and **findElements()** method.

Interview Question:

Difference between findElement and findElements

findElement	**findElements**
Returns a single web element	Returns a collection of web elements.
Returns the first matching web element within the web page even if multiple web elements match the locator value	Returns a list of multiple web elements matching the locator value.
Throws NoSuchElementException in case there are no matching elements.	Returns an empty list in case there are no matching elements.

1.Id: Id locator is used to locate a particular web element using the value of its 'id' attribute.

Syntax: driver.findElement(By.id ("element ID"))

2.Name: Name locator is used to locate a particular web element using the value of its 'name' attribute.

Syntax: driver.findElement(By.name("element ID"))

3.Class Name: Class Name locator is used to locate a particular web element using the value of its 'Class' attribute.

Syntax: driver.findElement(By.className("element ID"))

4.Tag Name: Tag Name locator is used to locate a particular web element using its Tag Name.

Syntax: driver.findElement(By.tagName ("htmltagname"))

5.LinkText: LinkText locator is used to locate a particular web element through its Link Text.

Syntax: driver.findElement(By.linkText ("linkText")

6.PartialLinkText: PartialLinkText locator is used to locate a particular web element using its partial Link Text.

Syntax: driver.findElement(By.partialLinkText ("Partaillinktext"))

7.CSS Selector: **CSS** stands for **Cascading Style Sheets** - CSS Selector in Selenium should be opted if you cannot locate an element using ID or Name locators. which identifies an element based on the combination of HTML tag, id, class and attributes.

- Once we start using CSS Selectors, we will love the speed when compared with Xpath.
- CSS Selector is the best way to locate complex elements in the web page.

Different types of CSS Selectors:

Tag and ID - We can use “**#**” notation to select the “**id**” attribute of an element.

Syntax: “tagname#<id value>”

```
    //Tag and ID (#)
    driver.findElement(By.cssSelector("input#gh-ac")).sendKeys
("Iphone 15 pro");
```

Tag and Class - In CSS, we can use “.” (dot) notation to select the “class” attribute of an element.

Syntax: “<tagname>.<class value>”

```
    //Tag and Class (.)
driver.findElement(By.cssSelector("button.gh-control")).click();
```

Tag and attribute: Apart from “**id**” and “**class**”, other attributes can also be used to locate web elements using CSS selector.

Syntax: “tagname[attribute=value]”

```
    //Tag and Attribute ((HTML Page)[Attribute=Value])
driver.findElement(By.cssSelector("input[type=\"submit\"]")).
click();
```

Tag, Class and attribute: This locator is used with the class name and other attribute values.

Syntax: “<tagname>.(class attribute value)[attribute = value]”

```
//Tag, class and Attributes
    driver.findElement(By.cssSelector("input.gh-tb
[role=\"combobox\"]")).clear();
```

Xpath:

- Xpath is a language used to locate web elements in xml document.
- XPath can be used as a substitute when you don't have a suitable id or name attribute for the element you want to locate.

  Syntax: findElement(By.xpath("XPath"));

  XPath = //tagname[@Attribute=’Value’]

Absolute Xpath –

- Absolute Xpath is the simplest form of XPath in Selenium. It starts with a single slash ‘/’ and provides the absolute path of an element right from the parent node.
- `/html/div/div/div/div[1]/a/img`

Relative Xpath –

- This XPath expression starts from the middle of the DOM structure. It is represented by a double slash '//' denoting the current node.
- It is always preferred over an absolute XPath as it is not a complete path from the root element.
- `driver.findElement(By.xpath("//input[@id='gh-ac']"))`

XPath using Contains:

Syntax: //tagname[contains(@attribute,constantvalue)]

Example: //button[contains(@class,'submit-btn ')]

XPath using Logical Operators: OR & AND

Syntax: OR //tagname[@attribute1=value1 OR @attribute2=value1]

Example: //input[@name="email" **or** contains(@placeholder,'abc')]

Syntax: AND //tagname[@attribute1=value1 AND @attribute2=value1]

Example: //input[@name='email' **and** contains(@placeholder,'Email')] .click();

XPath using Text():

Syntax: //tagname[text()='Text of the Web Element']

Example: //span[text()='Sign up with Google']

XPath using Starts-With()

Syntax: //tagname[starts-with(@attribute,value)]

Example: //input[starts-with(@placeholder,'Search')]

XPath using Index

Syntax: //tagname[@attribute='value'][Index Number]

Example: (//div[@class='form-group']//select)[2]

Chained XPath in Selenium

Syntax: //tagname1[@attribute1=value1]//tagname2[@attribute2=value2]

Example: //div[@id='gh-ac-box2']//input[@id='gh-ac']

Write Xpath in Selenium using Axes methods?

Axes methods are used when the exact element tagname or its attribute value is dynamic and cannot be used to locate an element. In such cases locating elements after traversing through child/sibling or parent becomes an easy approach.

Some of the widely used Xpath Axes are as below:

XPath using 'Following':
On using Following, It selects all the nodes that appear after the current node.
Syntax: //tagname[@attribute='value']//following::tagname
Example: //input[@id='gh-ac']//following::input

XPath using Following-sibling:
It selects all the nodes that have the same parent (at same level) as the current node and appear after the current node.
Syntax: //tagname[@attribute='value']//following-sibiling::tagname
Example: //div[@id='gh-ac-box2']//following-sibling::input

XPath using Preceding:
This method helps locate all the elements before the current node.
Syntax: //tagname[@attribute='value']//preceding::tagname
Example: //input[@id='gh-ac']//preceding::button[@id='gh-shop-a']

XPath using Child:
This approach is used to locate all the child elements of a particular node.
Syntax: //tagname[@attribute='value']//child::tagname
Example: //div[@id='gh-ac-box2']//child::input[@id='gh-ac']

XPath using Parent:
This method is used to select the parent node of the current node.
Syntax: //tagname[@attribute='value']/parent::tagname
Example: //span[@role='status']//parent::div//input[@id='gh-ac']

XPath using Descendants: [check later]
This selects all of the descendants(child, grand child, etc) of the current node.

Syntax: //tagname[@attribute=’value’]//descendants::tagname

Example: //div[@class='a-section a-spacing-base ap_email_fields']//descendant::div

XPath using Ancestors:

It selects all the ancestors(parent, grand parent, etc) of the current node.

Syntax: //tagname[@attribute=’value’]//ancestors::tagname

Example: //label[@for='ap_email']//ancestor::div

Synchronization

The Process of matching the execution speed of selenium with the execution speed of the application is called as Synchronization.

1. Implicit Wait
2. Explicit Wait
3. Fluent Wait

In **Implicit Wait**, we can tell selenium that we would like it to wait for a certain amount of time before throwing an exception that it cannot find the element on the page.
After the specified time, it will throw “No Such Element Exception”.
Syntax:

```
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(20));
```

The **Explicit Wait** in Selenium is used to tell the Web Driver to wait for certain conditions (Expected Conditions) or maximum time exceeded before throwing “ElementNotVisibleException” exception.

Syntax:

```
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(20));
wait.until(ExpectedConditions.visibilityOfElementLocated(By.name("btnK”")));
```

Fluent Wait – It is used to tell the WebDriver to wait for a certain condition (web element) becomes visible. It also defines how frequently(check the condition at the regular interval of time) WebDriver will check if the condition appears before throwing the “NoSuchElementException”.

Syntax:

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

fwait.until(ExpectedConditions.visibilityOfElementLocated(By.name("btnK")));
```

WaitExample:

```
public class Waits {

	public static void main(String[] args) {

	WebDriver driver = new ChromeDriver();
	driver.navigate().to("https://www.ebay.com/");
	driver.manage().window().maximize();

	//implicit wait
	driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(40));

	driver.findElement(By.xpath("//input[@id='gh-ac']")).sendKeys("Mobiles");
	driver.findElement(By.xpath("//input[@type='submit']")).click();

	//Explicit Wait
	WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(20));

	wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//select[@id='g
h-cat']")));

	//Fluent Wait
	Wait<WebDriver> fwait = new FluentWait<>(driver)
		.withTimeout(Duration.ofSeconds(20))
		.pollingEvery(Duration.ofSeconds(5))
		.ignoring(ElementNotInteractableException.class)
		.ignoring(NoSuchElementException.class);

		driver.findElement(By.xpath("//select[contains(@aria-label,'category')]")).click();

		driver.quit();
	}
}
```

Drop down in Selenium:

- The **'Select'** class in Selenium WebDriver is used for selecting and deselecting option in a dropdown.
- create the object of the ***Select*** class by using the following syntax.

Syntax: `Select select = new Select(WebElement webelement);`

- The *Select class constructor* accepts one parameter: the WebElement object returned by the locators of the specified element.

Select value from a dropdown:

- selectByIndex
- selectByValue
- selectByVisibleText

selectByIndex
This method selects the dropdown option by its index number. Here we provide an integer value as the index number as an argument. The index starts at **0**.

Syntax: `selectByIndex(int arg0)`

selectByValue
This method selects the *dropdown* option by its *value.* We provide a string value as the value as an argument.

Syntax: `selectByValue(String arg0)`

SelectByVisibleText
This method helps us to select one option from the dropdown or multi-select dropdown based on the dropdown text. We need to pass the String value as an argument.

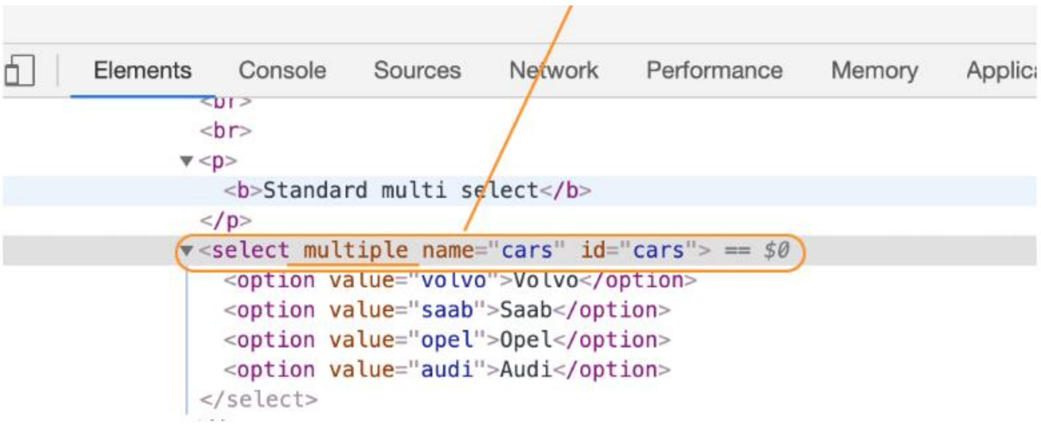
Syntax: `selectByVisibleText(String arg0)`

Example demonstrating all the 3 types:

```
public class Dropdown {

 public static void main(String[] args) throws Exception {

        WebDriver driver = new ChromeDriver();
        driver.navigate().to("https://www.ebay.com/");
        driver.manage().window().maximize();

        // implicit wait
                driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(20));

        // Handle drop down
        Select sel = new Select(driver.findElement(By.id("gh-cat")));

                // 1. Select by Visible Text
                m
                Thread.sleep(3000);

                // 2. Select by Value
                sel.selectByValue("625");
                Thread.sleep(3000);
                // 3. Select by index
                sel.selectByIndex(1);
                Thread.sleep(3000);

                driver.quit();
        }
}
```

Selecting multiple values from a dropdown in Selenium:

If the *<select >* tag contains **multiple** attributes, it means that the dropdown allows selecting multiple values as shown in below screenshot.



```
<br>
<br>
▼<p>
   <b>Standard multi select</b>
 </p>
▼<select multiple name="cars" id="cars"> == $0
   <option value="volvo">Volvo</option>
   <option value="saab">Saab</option>
   <option value="opel">Opel</option>
   <option value="audi">Audi</option>
 </select>
```

Check whether dropdown is Multi-Select?

The Select class provides the "isMultiple()" method, which determines whether the web element supports multiple selections. It returns a boolean value, i.e., True/False.

Syntax: isMultiple()

getOptions(): This method will get all the options of drop-down in List.

```
List<WebElement> list = sel.getOptions();
    System.out.println("All the Options are");

    for (WebElement i : list) {
        System.out.println(i.getText());
    }
```

getFirstSelectedOption(): This method will get the First Selected Option from Drop-down.

System.out.println("getFirstSelectedOption() = " + sel.getFirstSelectedOption().getText());

getAllSelectedOptions(): This method will get All the Selected Option from Drop-down.

```
List<WebElement> selList = sel.getAllSelectedOptions();
	System.out.println("All the Selected Options are");

	for (WebElement a : selList) {
		System.out.println(a.getText());
	}
```

Deselect: Just like Select values in a drop down and multi select, we can deselect the dropdown values too. Deselect method will work only for Multi-Select.

- deselectAll()
- deselectByIndex()
- deselectByValue()
- deselectByVisibleText()

Example Demonstrate all the methods:

```
public class DropDownMultiSelect {

	public static void main(String[] args) throws Exception {
		WebDriver driver = new ChromeDriver();
		driver.get("https://demoqa.com/select-menu");
		driver.manage().window().maximize();

		// implicit wait
		driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(20));

		// Multi select
		Select sel = new Select(driver.findElement(By.xpath("//*[@id='cars']")));
		if (sel.isMultiple()) {

			// Selecting multiple values by index
			sel.selectByIndex(1);
			sel.selectByIndex(2);

			// Or selecting by values
			sel.selectByValue("volvo");
			sel.selectByValue("audi");

			// Or selecting by visible text
			sel.selectByVisibleText("Saab");
			// sel.selectByVisibleText("Opel");
		}
		Thread.sleep(3000);
```

```
        // return true if multi-select dropdown is present
        Boolean b = sel.isMultiple();
        System.out.println("isMultiple: "+b); // true

        //Get all the options of drop-down in List
        List<WebElement> list = sel.getOptions();
        System.out.println("All the Options are");

        for (WebElement i : list) {
                System.out.println(i.getText());
        }

    //Get the First Selected Option from Drop-down
    System.out.println("getFirstSelectedOption() = " + sel.getFirstSelectedOption().getText());

        //Get All the Selected Option from Drop-down
        List<WebElement> selList = sel.getAllSelectedOptions();
        System.out.println("All the Selected Options are");

        for (WebElement i : selList) {
                System.out.println(i.getText());
        }

        // Deselect the options
        sel.deselectAll();
        sel.deselectByIndex(1);
        sel.deselectByValue("6");
        sel.deselectByVisibleText("Volvo");

        Thread.sleep(3000);

                driver.quit();
        }
}
```

Alerts in Selenium

Alerts are small popup boxes/windows which display the messages/notifications and notify the user with some information.

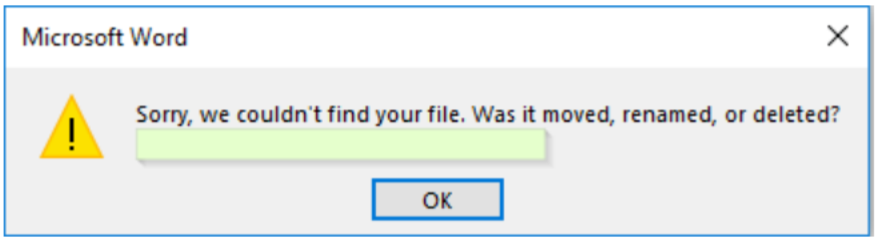
There are 3 types of alerts. And each of these alerts/popups needs different kinds of actions to handle these alerts.\

“?

1. Simple Alert:

- These alerts are just informational alerts and have an OK button on them.

- Users can click on the OK button after reading the message displayed on the alert box.

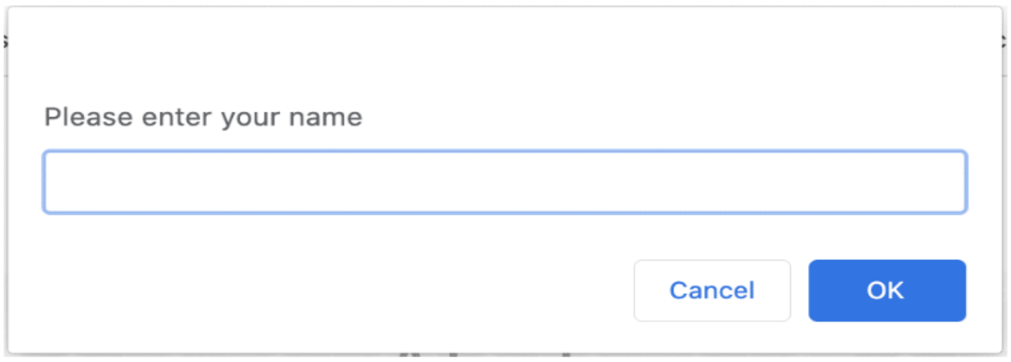

Microsoft Word

Sorry, we couldn't find your file. Was it moved, renamed, or deleted?

2. Prompt Alert:

- In these alerts users are requires to provide some input in the alert box.

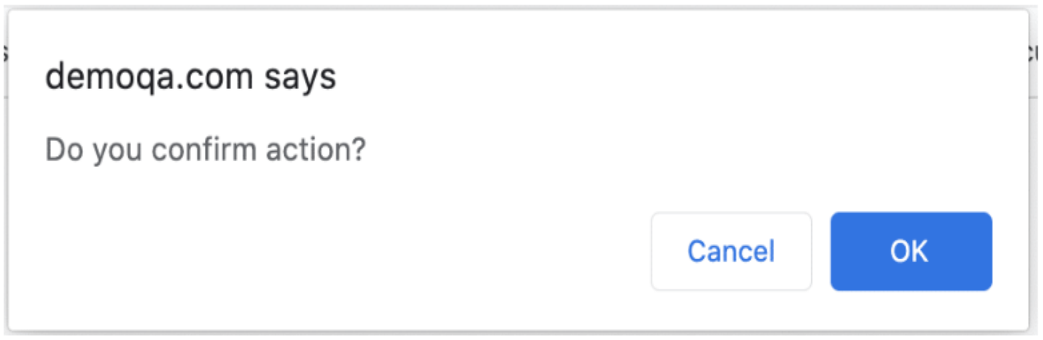
whenever an alert appears, it opens up a new window/popup. So, for handling the Alerts using Selenium WebDriver, we need to shift the control to the child window/popup opened by the Alerts. To switch the control from the parent window to the Alert window, Selenium WebDriver provides the following command to handle the Alerts.


Please enter your name

3. Confirmation Alert:

- These alerts get some confirmation from the user in the form of accepting or dismissing the message box.

- In these alerts users can only read the message and provide the inputs by pressing either OK or Cancel button.


demoqa.com says

Do you confirm action?

```
Syntax: Alert alt = driver.switchTo().alert();
```

To handle such alerts, Selenium provides the following methods:

1. accept(): This method clicks on the 'OK' button of the alert box.

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

2. dismiss(): This method is used to click on Cancel button on the alert.

```
driver.switchTo( ).alert( ).dismiss();
```

3. getText(): This method captures the message from the alert box.

```
driver.switchTo().alert().getText();
```

4. sendKeys("String args"): This method is used to send some data to the alert box.

```
driver.switchTo().alert().sendKeys("Text");
```

Example #1:

```
 public class AlertClassOne {
	public static void main(String[] args) throws Exception {
		WebDriver driver = new ChromeDriver();
		driver.get("https://demoqa.com/alerts");
		driver.manage().window().maximize();

		JavascriptExecutor js = (JavascriptExecutor) driver;
		// It will scrolldown the page by 300-pixel vertical
		js.executeScript("window.scrollBy(0,300)");

		driver.findElement(By.id("confirmButton")).click();

		// Switching to Alert
		Alert alert = driver.switchTo().alert();

		// Capture & display alert message.
		String alertMessage = alert.getText();
		System.out.println(alertMessage);

		// clicks on the 'OK' button of the alert box
		alert.accept();
		driver.quit();
	}
}
```

Example #2:

```
public class AlertClassTwo {
	public static void main(String[] args) throws Exception {
		WebDriver driver = new ChromeDriver();
		driver.get("https://demoqa.com/alerts");
		driver.manage().window().maximize();

		JavascriptExecutor js = (JavascriptExecutor) driver;
		// This will scroll down the page by 1000 pixel vertical
		js.executeScript("window.scrollBy(0,300)");

		driver.findElement(By.id("promtButton")).click();

		// Switching to Alert
		Alert alert = driver.switchTo().alert();

		//Enter text in alert text field
		driver.switchTo().alert().sendKeys("This is Prompt Alert");

		// clicks on the 'Cancel' button of the alert box
		alert.dismiss();
		driver.quit();
	}
```

}

Handle Pop-up window in Selenium (Window Handles)

In automation there are scenarios where multiple windows open within an application. And we need to handle those opened windows by switching the control among parent and child windows.

Two Imp Methods to handle Popup windows in selenium:

1. **getwindowhandle()** - is useful to get handle of current selected window in selenium.
2. **getWindowHandles()** – this will return handles of all open windows or tabs.

Both these methods **getwindowhandle()** and **getWindowHandles()** in selenium are useful when we are working with multiple windows and we need to switch from one window to another window.

#1 Example: getwindowhandle() method

```
public class WindowHandle {

  public static void main(String[] args) {

      WebDriver driver = new ChromeDriver();
      driver.get("https://www.ebay.com/");
      driver.manage().window().maximize();

      //Get current browser window-handle using getWindowHandle
      String currentWindow = driver.getWindowHandle();
      System.out.println(currentWindow +" :  is current browser
window handle");
      driver.quit();
      }
}
```

Output: `A3C7910FE181688C8A22309DB2254FDE :  is current browser window handle`

In above example you can see that selenium **getwindowhandle()** method has retrieved current browser's window handle and print it in console.

#2 Example: getwindowhandles() method in selenium - is used to get all window handles.

```
public class WindowHandles {

  public static void main(String[] args) throws Exception {

    WebDriver driver = new ChromeDriver();
    driver.get("https://only-testing-blog.blogspot.com/2014/01/textbox.html");
    driver.manage().window().maximize();

    driver.findElement(By.xpath("//b[contains(text(),'Open New Page')]")).click();

    //Get main window handle using getWindowHandle()
    String parentWindow = driver.getWindowHandle();
    System.out.println("Current window handle is = "+parentWindow);

    //Get all windows handles using getWindowHandles()
    Set<String> allHandles=driver.getWindowHandles();
    System.out.println("Total windows number = "+allHandles);

    String childWindow = null;
    //Iterate through allHandles
        for (String handle : allHandles) {
        //If not match with parent window then it is child tab handle
        //and store it in childWindow variable
        if(!parentWindow.equals(handle)) {
                childWindow = handle;
                System.out.println("Second Window handle is = "+childWindow);
                }
        }

        //Switch to tab two of browser using childWindow
        driver.switchTo().window(childWindow);

        //type text in text box of child tab
        driver.findElement(By.id("fname")).sendKeys("Subhash Java");

        //Switch to tab one of browser using parentWindow
        driver.switchTo().window(parentWindow);
        //driver.swithTo().defaultContent(); //This will also take the control to the parent page
        //select radio button on tab one.
        driver.findElement(By.id("radio2")).click();

        driver.quit();

        }
}
```

How to get all available Links on the page?

```
public class AllTheLinks {

	public static void main(String[] args) {
		WebDriver driver = new ChromeDriver();
		driver.get("https://www.ebay.com/");
		driver.manage().window().maximize();

		List<WebElement> allLinks = driver.findElements(By.tagName("a"));
		System.out.println("Link count = "+allLinks.size());

		for(WebElement link : allLinks) {
			System.out.println(link.getText());
		}
		driver.quit();
	}
}
```

How to get all the Broken Links?

We should always make sure that there are no broken links on the site because the user should not land into an error page.

For checking the broken links, we will need to do the following steps.

- Collect all the links in the web page based on <a> tag.
- Send HTTP request for the link and read HTTP response code.
- Find out whether the link is valid or broken based on HTTP response code.
- Repeat this for all the links captured.

```
public class brokenLinks {

  public static void main(String[] args) {
	WebDriver driver = new ChromeDriver();
	driver.get("http://www.deadlinkcity.com/");
	driver.manage().window().maximize();
	driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

	//find hyperlink
	List<WebElement> allLinks = driver.findElements(By.tagName("a"));
	System.out.println("Total available links = "+allLinks.size());

	int resCode = 200; //for valid link
	int brokenLinkCount = 0;

	//Iterate all the links to get the url using getAttribute=href
```

```
        for(WebElement element : allLinks) {
        String url = element.getAttribute("href");
        try {
                //Open the url using HttpURLConnection
                URL urlLink = new URL(url);
                HttpURLConnection huc = (HttpURLConnection) urlLink.openConnection();
                //sending the request to the link
                huc.setRequestMethod("HEAD");
                //connect
                huc.connect();
                //check the response code
                resCode = huc.getResponseCode();

                //If the response code is 400 or above, print the broken url
                if(resCode >= 400) {
                System.out.println(url+ " : is broken link");
                                        brokenLinkCount++;
                                }

                        }catch(Exception e) {
                                e.printStackTrace();
                        }
                }
                System.out.println("Total broken Links = "+brokenLinkCount);
                driver.quit();
        }
}
```

Action Class in Selenium

There are situations, where we have to automate complex interactions like **Drag-n-Drop** and **Double-click** which cannot be done by simple **WebElement** commands. So, to handle those types of advance actions we have the **Actions** class in Selenium.

build(): method is used to compile all the list of actions into a single step and ready to be performed.
perform(): Use the perform() method when executing the Action object
builder: builder pattern builds a Composite Action containing all actions specified by the method calls.

Syntax: Actions act = new Actions(driver)

Handling Keyboard and Mouse events

1. **Methods for performing Keyboard Events:**

keyDown(modifier key): Performs a key press.

sendKeys(keys to send): Sends keys to the web element.

keyUp(modifier key): Performs a key release.

2. **Different Methods for performing Mouse Events:**

click(): Clicks at the current mouse location.

doubleClick(): Performs a double-click at the current mouse location.

contextClick(): Performs a context-click at middle of the given element (**Right Click**).

clickAndHold(): Clicks (without releasing) in the middle of the given element.

dragAndDrop(source, target): Click-and-hold at the location of the source element, moves to the location of the target element.

dragAndDropBy(source, xOffset, yOffset): Click-and-hold at the location of the source element, moves by a given offset.

moveByOffset(x-offset, y-offset): Moves the mouse from its current position (or 0,0) by the given offset.

moveToElement(toElement): Moves the mouse to the middle of the element (**Mouse hover**)

release(): Releases the left mouse button at the current mouse location.

Example #1: **`MouseHover`**

```
public class MouseHover {

	public static void main(String[] args) throws Exception {
		WebDriver driver = new ChromeDriver();
		driver.get("https://testsigma.com/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

		driver.findElement(By.id("hs-eu-confirmation-button")).click();
		Thread.sleep(3000);

		WebElement link_Home = 
driver.findElement(By.xpath("//*[text()='Resources']"));
		//Instantiate Action Class
		Actions actions = new Actions(driver);
		actions.moveToElement(link_Home);
		actions.perform();
		driver.quit();
	}
}
```

Example #2: **`Right Click`**

```
public class RightCLick {

	public static void main(String[] args) throws Exception {
		WebDriver driver = new ChromeDriver();
		driver.get("https://demo.guru99.com/test/simple_context_menu.html");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

		Actions action = new Actions(driver);
		WebElement rtClick = driver.findElement(By.xpath("//span[text()='right click 
		me']"));

		action.contextClick(rtClick).perform();
		WebElement element = driver.findElement(By.cssSelector(".context-menu-
		icon-copy"));
		element.click();

		//Accept the alert displayed
		driver.switchTo().alert().accept();
		driver.quit();
	}
}
```

Example #3: **Double Click**

```
public class DoubleClick {

	public static void main(String[] args) throws Exception {
		WebDriver driver = new ChromeDriver();
		driver.get("https://demo.guru99.com/test/simple_context_menu.html");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

		Actions action = new Actions(driver);
		WebElement doubleClick =
		driver.findElement(By.xpath("//button[text()='Double-Click Me To See
		Alert']"));

		action.doubleClick(doubleClick).perform();

		//Switch to the alert box and click on OK button
		Alert alert = driver.switchTo().alert();
		System.out.println("Alert Text\n" +alert.getText());
		alert.accept();

		driver.quit();
	}
}
```

Example #4: **DragAndDrop**

```
public class DragAndDrop {

	public static void main(String[] args) throws Exception {
		WebDriver driver = new ChromeDriver();
		driver.get("https://demo.guru99.com/test/drag_drop.html");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

		Actions action = new Actions(driver);

		//Element which needs to drag
		WebElement drag = driver.findElement(By.xpath("//*[@id='credit2']/a"));

		//Element on which need to drop
		WebElement drop = driver.findElement(By.xpath("//*[@id='bank']/li"));
		action.dragAndDrop(drag, drop).perform();

		driver.quit();
	}
}
```

Example #5: **KeyBoard Events**

```
public class KeyBoardEvents {
	public static void main(String[] args) throws Exception {

	WebDriver driver = new ChromeDriver();
	driver.get("https://demoqa.com/text-box");
	driver.manage().window().maximize();
	driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

	// Create object of the Actions class
	Actions actions = new Actions(driver);

	// Enter the Current Address
	WebElement currentAddress = driver.findElement(By.id("currentAddress"));
	currentAddress.sendKeys("1212, HSR Layout Sector-1");

	// Copy the Current Address using CTRL + A
	actions.keyDown(Keys.CONTROL);
	actions.sendKeys("a");
	actions.keyUp(Keys.CONTROL);
	actions.perform();

	// Copy the Current Address using CTRL + C
	actions.keyDown(Keys.CONTROL);
	actions.sendKeys("c");
	actions.keyUp(Keys.CONTROL);
	actions.perform();

	// Press the TAB Key to move to Permanent Address text field
	actions.sendKeys(Keys.TAB);
	actions.perform();

	// Paste the Address in the Permanent Address field using CTRL + V
	actions.keyDown(Keys.CONTROL);
	actions.sendKeys("v");
	actions.keyUp(Keys.CONTROL);
	actions.perform();

	driver.quit();

	}
}
```

Example #6: **SeriesOfActions**

```
public class SeriesOfActions {

	public static void main(String[] args) throws Exception {
		WebDriver driver = new ChromeDriver();
		driver.get("https://www.facebook.com/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

		Actions actns = new Actions(driver);
		WebElement uName = driver.findElement(By.id("email"));

		Action seriesOfActions=actns.moveToElement(uName)
						.click()
						.keyDown(uName, Keys.SHIFT)
						.sendKeys("subhash")
						.keyUp(uName, Keys.SHIFT)
						.doubleClick(uName)
						.contextClick()
						.build();

		seriesOfActions.perform();

		driver.quit();
	}
}
```

How to Scroll Down or UP Page in Selenium?

To scroll using Selenium, we can use JavaScriptExecutor interface that helps to execute JavaScript methods through Selenium Webdriver.

Syntax:

```
JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript(Script,Arguments);
```

To scroll using Selenium, we can use JavaScriptExecutor interface that helps to execute JavaScript methods through Selenium Webdriver.

Scenario 1: To scroll down the web page by pixel.

```
public class ScrollDown1 {

	public static void main(String[] args) throws Exception {
		WebDriver driver = new ChromeDriver();
		driver.get("https://demoqa.com/alerts");
		driver.manage().window().maximize();
```

```
                    JavascriptExecutor js = (JavascriptExecutor) driver;
                    // It will scrolldown the page by 300-pixel vertical
                    js.executeScript("window.scrollBy(0,300)");

                    driver.findElement(By.id("confirmButton")).click();
                    driver.quit();
          }
}
```

Scenario 2: To scroll down the web page by the visibility of the element.

```
public class ScrollDown1 {

   public static void main(String[] args) throws Exception {

          WebDriver driver = new ChromeDriver();
          driver.get("http://demo.guru99.com/test/guru99home/");
          driver.manage().window().maximize();

          JavascriptExecutor js = (JavascriptExecutor) driver;

          //Find element by link text and store in variable "Element
          WebElement Element = driver.findElement(By.linkText("Linux"));

          // It will scroll down until the mentioned element is visible on the current page
          js.executeScript("arguments[0].scrollIntoView();", Element);

          /*
          //This will scroll the web page till end.
          js.executeScript("window.scrollTo(0,document.body.scrollH
          eight)");
          */

          driver.quit();
      }
}
```

isDisplayed, isSelected, isEnabled in Selenium

WebDriver facilitates the user with the following methods to check the visibility of the web elements. These web elements can be buttons, drop boxes, checkboxes, radio buttons, labels, etc.

1. **isDisplayed()** - determines if an element is displayed or not.
2. **isSelected()** - determines if an element is selected or not. widely used on checkboxes, radio buttons.

3. **isEnabled()** – determines if an element is enabled or not.
Example #1:

```
public class DisplayedEnabledSelected {

  public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        driver.get("https://duckduckgo.com/");
        driver.manage().window().maximize();

        // Verify that the "Search" Box is displayed
        WebElement searchBox =
        driver.findElement(By.className("searchbox_input__bEGm3"));
        if (searchBox.isDisplayed()) {
                System.out.println("Search Box is visible. Return: " +
                searchBox.isDisplayed());
        } else {
                System.out.println("Search Box is not visible. Return: " +
                searchBox.isDisplayed());
                }

        // Verify that the "Search" Box is enabled
        if (searchBox.isEnabled()) {
                System.out.println("Search Box is enabled. Return: " + searchBox.isEnabled());
                searchBox.sendKeys("Selenium");
        } else {
                System.out.println("Search Box is not enabled. Return: " +
                searchBox.isEnabled());
                }
        }
}
```

Example #2: - isSelected()- radio button handling.

```
public class IsSelected {

  public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        driver.get("https://demoqa.com/radio-button");
        driver.manage().window().maximize();

        List<WebElement> RadioOptions = driver.findElements(By.cssSelector(".custom-
        radio"));
        for(WebElement options: RadioOptions) {
                System.out.println("Options :"+ options.getText());
        }

        boolean sel = RadioOptions.get(0).isSelected();
        System.out.println(sel);
        driver.quit();
  }
```

}
Handle Cookies in Selenium WebDriver

- A HTTP cookie is comprised of information about the user and their preferences. It stores information using a key-value pair.
- It is a small piece of data sent from Web Application and stored in Web Browser, while the user is browsing that website.

```
driver.manage().deleteAllCookies();
```

TestNG in Selenium

- TestNG is one of the most widely used open-source automated **testing framework** used in automation testing suite for developing the selenium framework.

- TestNG provides us the control over the test cases and execution of the test cases.

- TestNG framework came after Junit, and TestNG adds more powerful functionality and easier to use.

- TestNG provides powerful and flexible test cases with help of easy annotations, grouping, sequencing and parametrizing.

- Using TestNG, we can generate a proper report, and we can easily come to know how many test cases are passed, failed, and skipped. We can execute the failed test cases separately.

- TestNG framework allows us to define the test cases where each test case is independent of other test cases.

There are three major advantages of TestNG over JUnit:

- Annotations are easier to understand
- Test cases can be grouped more easily
- Parallel testing is possible

What is Annotation in TestNG

Annotations in TestNG are lines of code that can control how the method will be executed. TestNG annotations are always represented by @ symbol.

Example: @Test

Setup a new TestNG Project in Eclipse

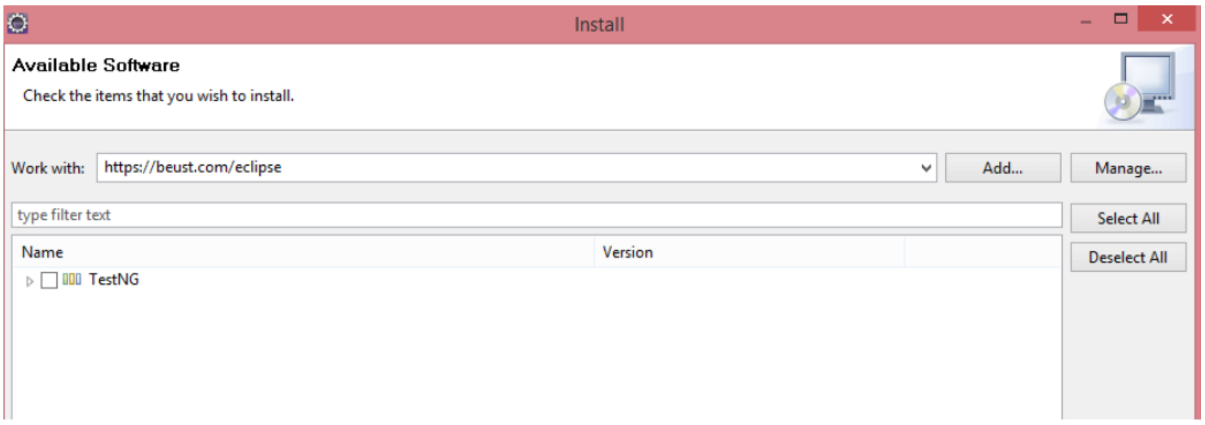
Step1. Go to the official website of the TestNG. Click on the link given below: https://testng.org/doc/

Step2. Click on the **Eclipse link** appearing on the top menu bar.

Step3. Click on the **Installation link**, and then click on the "**install the plug-in**". It will take us to the screen where the URL is given https://testng.org/testng-eclipse-update-site under the category "Install site for release". To install the TestNG plug-in in Eclipse, we need to add this given URL in Eclipse.

Step4. Open the Eclipse. Click on the **Help** appearing on the menu bar and then click on the **Install New Software**

Step5. Copy the URL https://testng.org/testng-eclipse-update-site and press the Enter, we will see **Pending** for few seconds, thereafter we will see that **TestNG** plug-in has been loaded.



Step6. Click on the TestNG checkbox and click on Next. This will install all the dependencies of the TestNG.

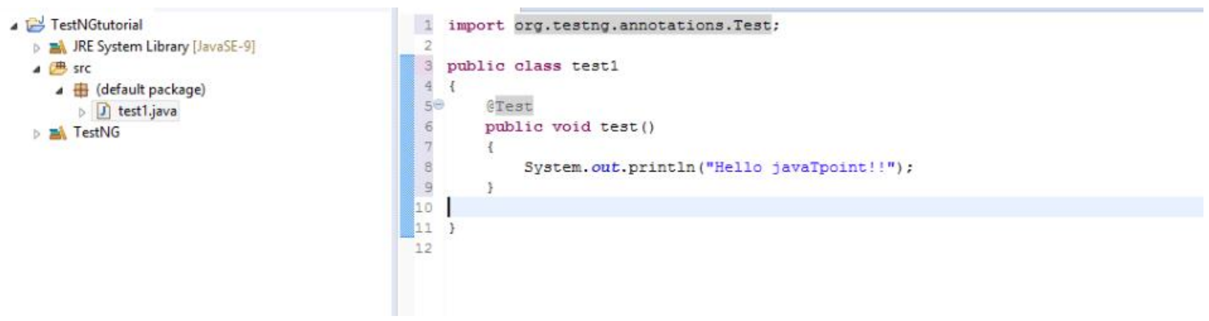
Step7. Accept the license and then click on the Finish.

Running test cases in TestNG without java compiler

Step 1: Create a TestNG project. It is created in the same way as we create the java project. Click on the **File > New > Java project**.

Step 2: Create Class file in a src folder

Step 3: If we install TestNG framework or TestNG library, then we do not need to compile on Java compiler (no need to add main method to execute program) as TestNG itself is a java compiler that compiles the test cases. In order to achieve this, we need to install the TestNG plug-in and then add the TestNG plug-in in a project.


TestNGtutorial
JRE System Library [JavaSE-9]
src
(default package)
test1.java
TestNG

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

public class test1
{
    @Test
    public void test()
    {
        System.out.println("Hello javaTpoint!!");
    }

}
```

```
public class FirstTestNGProgram {

	@Test
	public void Test() {
		System.out.println("First TestNG Program");
	}
}
```

Importance of XML file in TestNG Configuration

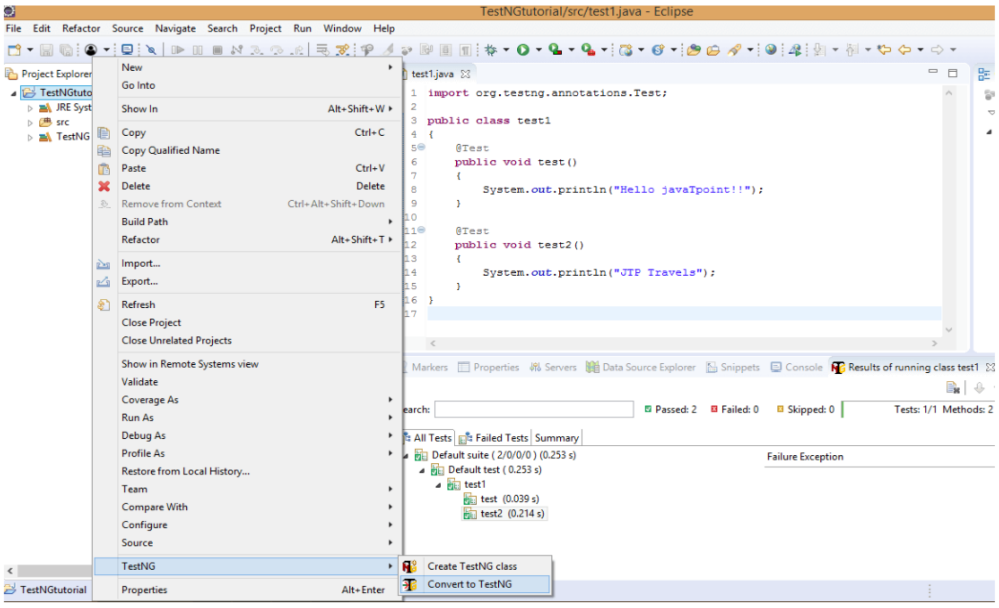
In TestNG, you can define multiple test cases in a single class whereas, in Java, wew can define only one test in a single class in the main() method.
In Java, if we want to create one more test, then we need to create another java file and define the test in the main() method.
Instead of creating test cases in different classes, we use TestNG framework that allows us to create multiple test cases in a single class.

Example:

```
	@Test
	public void Test1() {
		driver = new ChromeDriver();
		driver.get("https://www.ebay.com/");
		driver.manage().window().maximize();
	}

	@Test
	public void Test2() {
		System.out.println(driver.getTitle());
		System.out.println("Current url is : " + driver.getCurrentUrl()
	}

	@Test
	public void Test3() {
		driver.findElement(By.id("gh-ac")).sendKeys("HP Laptop");
		driver.findElement(By.id("gh-btn")).click();
		System.out.println(driver.getTitle());
		driver.quit();
	}
```

How to create a xml file

- Right click on the project. Move the cursor down, and we will see TestNG and then click on the Convert to TestNG.



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

public class test1
{
    @Test
    public void test()
    {
        System.out.println("Hello javaTpoint!!");
    }

    @Test
    public void test2()
    {
        System.out.println("JTP Travels");
    }
}
```

```
Passed: 2    Failed: 0    Skipped: 0    Tests: 1/1  Methods: 2
Default suite ( 2/0/0/0 ) (0.253 s)
  Default test ( 0.253 s)
    test1
      test  (0.039 s)
      test2  (0.214 s)
```

- Click on Next on preview of the xml page >> click on Finish

- The testing.xml file is created as shown below:

```
https://testng.org/testng-1.0.dtd (doctype)
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite">
  <test thread-count="5" name="Test">
    <classes>
      <class name="testNG.FirstTestNGProgram"/>
      <class name="testNG.FirstTestNGProgram2"/>
    </classes>
  </test> <!-- Test -->
</suite> <!-- Suite -->

```

- Now we do not need to run the java files individually. We have to run the XML file as TestNG suite which will automatically execute all the test from all the class files specified inside the XML file that are containing test cases.

Include or exclude test cases:

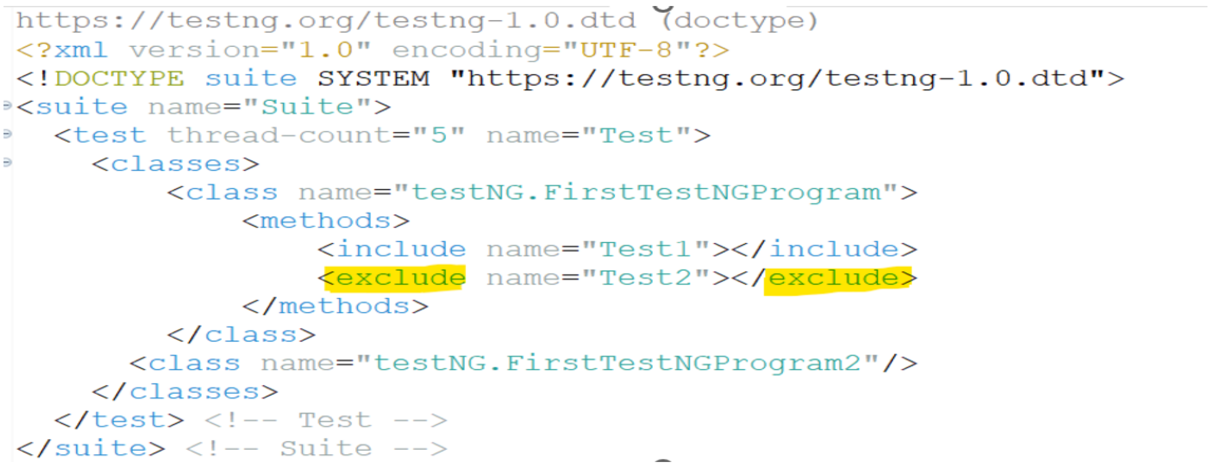
1. Use enabled=false to exclude the test case in the class file only

```
@Test(enabled = false)
public void Test1() {
        System.out.println("Test1");
}
```

```
	@Test
	public void Test2() {
		System.out.println("Test2");
	}
```

Output: Here only Test2 will run and Test1 will be skipped as we have disabled the test case.

2. Include/exclude the test cases from testing.xml file like shown below:



```
https://testng.org/testng-1.0.dtd (doctype)
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite">
  <test thread-count="5" name="Test">
    <classes>
        <class name="testNG.FirstTestNGProgram">
            <methods>
                <include name="Test1"></include>
                <exclude name="Test2"></exclude>
            </methods>
        </class>
      <class name="testNG.FirstTestNGProgram2"/>
    </classes>
  </test> <!-- Test -->
</suite> <!-- Suite -->
```

In the above file, Test2 will be skipped as we have excluded the test case.

TestNG Groups:

TestNG Groups allow us to perform groupings of different test methods. Grouping of test methods is required when we want to access the test methods of different classes.

Groups are specified in the testng.xml file with <groups> tag.

```
public class FirstTestNGProgram
	@Test(groups= {"SmokeTest"})
	public void Test1() {
		System.out.println("Test1");
	}

	@Test
	public void Test2() {
		System.out.println("Test2");
	}
public class FirstTestNGProgram2
	@Test
	public void Test3() {
		System.out.println("Test3");
	}
	@Test(groups= {"SmokeTest"})
	public void Test4() {
		System.out.println("Test4");
	}
```

Below mentioned is the syntax of how to declare groups in an XML file e.g.

```
<suite name="Suite">
        <test thread-count="5" name="Test">
                <groups>
                        <run>
                                <include name="SmokeTest" />
                        </run>
                </groups>
                <classes>
                        <class name="testNG.FirstTestNGProgram">
                                <methods>
                                        <include name="Test1"></include>
                                        <exclude name="Test2"></exclude>
                                </methods>
                        </class>
                        <class name="testNG.FirstTestNGProgram2" />
                </classes>
        </test> <!--Test -->
</suite> <!--Suite -->
```

TestNG Priority:

We write Priority parameter so that each test should run against to the priority assigned to them.

Example:

```
        @Test(priority = 2)
        public void Test1() {
                System.out.println("Test1");
        }

        @Test(priority = 1)
        public void Test2() {
                System.out.println("Test2");
        }
```

In above example, Test2 will run first and Test1 will run at the end because of priority assigned to the test cases.

TestNG Annotations

TestNG Annotation is a piece of code which is used to control the flow of execution of test methods.

@BeforeSuite	The @BeforeSuite annotated method will run before the execution of all the test methods in the suite.
@AfterSuite	The @AfterSuite annotated method will run after the execution of all the test methods in the suite.
@BeforeTest	The @BeforeTest annotated method will be executed before the execution of all the test methods of available classes belonging to that folder.
@AfterTest	The @AfterTest annotated method will be executed after the execution of all the test methods of available classes belonging to that folder.
@BeforeClass	The @BeforeClass annotated method will be executed before the first method of the current class is invoked.
@AfterClass	The @AfterClass annotated method will be invoked after the execution of all the test methods of the current class.
@BeforeMethod	The @BeforeMethod annotated method will be executed before each test method will run.
@AfterMethod	The @AfterMethod annotated method will run after the execution of each test method.
@BeforeGroups	The @BeforeGroups annotated method run only once for a group before the execution of all test cases belonging to that group.
@AfterGroups	The @AfterGroups annotated method run only once for a group after the execution of all test cases belonging to that group.

description

```
@Test(description="This is testcase1")
```

dependsOnMethods

```
	@Test(dependsOnMethods= {"WebStudentLogin"})
```

TestNG DataProviders

- Home work – Practice How to read data from excel.

Read data and Write data from/to Properties file:

Read from Properties file:

```
public class ReadFromPropertiesFile {

	public static void main(String[] args) throws IOException {

		String filePath = "D:\\Java-Selenium
		Course\\Automation\\SeleniumCourse\\testData\\TestData.properties";

		FileInputStream fis = new FileInputStream(filePath);
		Properties prop = new Properties();
		prop.load(fis);

		System.out.println("Url is : "+prop.getProperty("url"));
		System.out.println(prop.getProperty("email"));
		System.out.println(prop.getProperty("password"));
	}
}
```

Write to Properties file:

```
public class WriteToPropertiesFile {

	public static void main(String[] args) throws IOException {
		String filePath = "D:\\Java-Selenium Course\\Automation\\SeleniumCourse"
					+ "\\testData\\TestDataOut.properties";

		FileOutputStream fos = new FileOutputStream(filePath);
		Properties prop = new Properties();

		prop.setProperty("username","Subhash@gmail.com");
		prop.setProperty("password","Pass@123");
		prop.store(fos, "writing to a file");
	}
}
```

Read Data from Excel File:

```
public class ReadExcel {

	private static final int STRING = 0;
	private static final int NUMERIC = 0;

	@SuppressWarnings("deprecation")
	public static void main(String[] args) throws IOException {

		String excelPath = "D:\\Java-Selenium Course\\Automation"
					+ "\\SeleniumCourse\\testData\\ExcelFile.xlsx";

		FileInputStream file = new FileInputStream(excelPath);
		XSSFWorkbook wb = new XSSFWorkbook(file);
		XSSFSheet sheet = wb.getSheetAt(0);

		int rowsCount = sheet.getLastRowNum();
		int colCount = sheet.getRow(1).getLastCellNum();

		for(int i=0; i<=rowsCount; i++) {
			XSSFRow row = sheet.getRow(i);
			for(int j=0; j<colCount; j++) {
				XSSFCell cell = row.getCell(j);

				switch(cell.~~getCellType~~()) {
				case Cell.~~CELL_TYPE_STRING~~:
					System.out.print(cell.getStringCellValue()+ "\t");
					break;
				case Cell.~~CELL_TYPE_NUMERIC~~:
					System.out.print(cell.getNumericCellValue()+ "\t");
					break;
				}
				System.out.print(" | ");
			}
			System.out.println();
		}
	}
}
```

Assertions in TestNG

Assertions in TestNG are a way to verify that the expected result and the actual result matched or not. And this is to determine whether our test cases passed or failed.

TestNG asserts (or assertions) popularly validate the results in TestNG using selenium.

Types of Asserts:

1. Hard Assert
2. Soft Assert

1. **Hard Asserts** are those asserts that stop the test execution when an assert statement fails, and the subsequent assert and other statements are therefore not validated.

Example: Hard asserts are used where the login functionality is failing, there is no point in validating other pages which opens only when login works. **Hard asserts are the default type of asserts in TestNG we regularly use.**

Syntax:

```
    Assert.assertEquals(actualText, expectedText);
```

Example:

```
@Test
  public void login(){

    driver.findElement(By.xpath("//span[text()='My Account']")).click();
    driver.findElement(By.linkText("Login")).click();
    driver.findElement(By.id("input-email")).sendKeys(prop.getProperty("email"));
    driver.findElement(By.id("input-password")).sendKeys(prop.getProperty("password"));
    driver.findElement(By.xpath("//input[@type='submit']")).click();

    String actualText =
    driver.findElement(By.linkText("Edit your account  information")).getText();
    String expectedText = "Edit your account information";

    System.out.println("Before Assert");

    //Assertion is added below
    //If this assertion fails then
    //no further lines of code will be executed (example here – 2 print statements will not run)
    Assert.assertEquals(actualText,expectedText); //Validation
    System.out.println("Actual Text : "+actualText);
    System.out.println("Expexted Text : "+expectedText);
}
```

In the above example, Assert.*assertEquals*(actualText,expectedText) line will validate two strings and passes the test case if actualText is same as expectedText.
If there is no match, then the last two print statements will not be executed.

2. **Soft Assert in TestNG -** Soft asserts are just the opposite of hard asserts. In soft asserts, the subsequent assertions keep on running even though one assert validation fails, i.e., the test execution does not stop.

So, when should we use soft asserts in TestNG? - We use soft asserts when we do not care about the failure of specific validations and want the test execution to proceed and also want to see the exception errors.

Syntax:

```
SoftAssert softasrt = new SoftAssert();
softasrt.assertEquals(actualText, expectedText);
```

Example:

```
@Test
public void login() throws InterruptedException {
        driver.findElement(By.xpath("//span[text()='My Account']")).click();
        driver.findElement(By.linkText("Login")).click();
        driver.findElement(By.id("input-email")).sendKeys(prop.getProperty("email"));
        driver.findElement(By.
                        id("input-password")).sendKeys(prop.getProperty("password"));
        driver.findElement(By.xpath("//input[@type='submit']")).click();
        String actualText = driver.findElement(By.
                        linkText("Edit your account information")).getText();
        String expectedText = "Edit your account information";
        System.out.println("Before Assert");

        //Soft Assertion is added below
        //If the assertion fails then also
        //the below lines of code will be executed
        SoftAssert softassert = new SoftAssert();
        softassert.assertEquals(actualText, expectedText); //Validation

        System.out.println("Actual Text : "+actualText);
        System.out.println("Expected Text : "+expectedText);
}
```

In the above example, Soft assert - SoftAssert softassert = **new** SoftAssert() is added. Even if the assertion fails, still the last two print statements will run.

Commonly used TestNG Assert Methods:

Assert.assertEqual(String actual, String expected): This method validates if the actual and expected values are the same or not.

Assert.assertEqual(String actual, String expected, String message): Similar to the previous method just that when the assertion fails, the message displays along with the exception thrown.

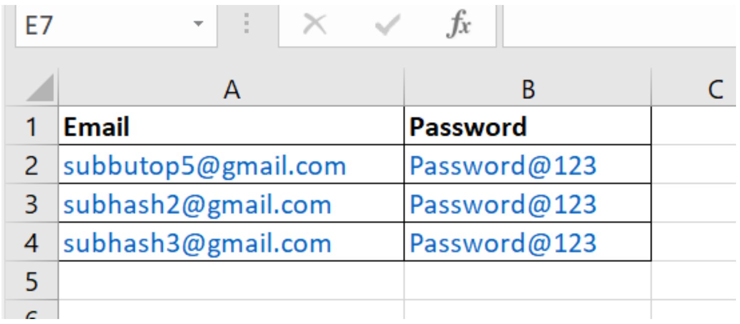
Assert.assertTrue(condition): This method asserts if the condition is true or not. If not, then the exception error is thrown.

Assert.assertTrue(condition, message): Similar to the previous method with an addition of message, which is shown on the console when the assertion fails along with the exception.

DataProvider in TestNG:

To perform data provider in Selenium do the following steps:

1. Create excel file “ExcelFile.xlsx”



E7

	A	B	C
1	Email	Password	
2	subbutop5@gmail.com	Password@123	
3	subhash2@gmail.com	Password@123	
4	subhash3@gmail.com	Password@123	
5			

2. Create a Java Class “ReadExcel.java” in some other package and Write below Java code to read the data from Excel file and return the data into 2-dimensional Object Array.

```
public class ReadExcel {

    // Read the data from Excel
    public static Object[][] getTestDataFromExcel(String sheetName) {

            XSSFWorkbook workbook = null;
            File file = new File("D:\\Java-Selenium
Course\\Automation\\SeleniumCourse\\testData\\ExcelFile.xlsx");

            try {
                    FileInputStream fisExcel = new FileInputStream(file);
                    workbook = new XSSFWorkbook(fisExcel);
            } catch (Throwable e) {
                    e.printStackTrace();
            }

            XSSFSheet sheet = workbook.getSheet(sheetName);

            int rows = sheet.getLastRowNum();
            int cols = sheet.getRow(0).getLastCellNum();

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

            for (int i = 0; i < rows; i++) {
                    XSSFRow row = sheet.getRow(i + 1);
                    for (int j = 0; j < cols; j++) {
                            Cell cell = row.getCell(j);

                            if (cell != null) {
                                    if (cell.getCellTypeEnum() == CellType.STRING) {
                                            data[i][j] = cell.getStringCellValue();
                                            System.out.print(data[i][j]+"\t");
                                    } else if (cell.getCellTypeEnum() ==
    CellType.NUMERIC) {
                                            data[i][j] = cell.getNumericCellValue();
                                            System.out.print(data[i][j]+"\t");
                                    }
                            } // main if
                            System.out.print(" | ");
                    }
                    System.out.println();
            }

            return data;
        }
    }
```

Create @Test Method in "LoginTestClassWork.java" class and create your test case for login functionality and also create @DataProvider(name="supplyingTestData") method to bring the data from excel to the @Test selenium code.

```
public class LoginTestClassWork {

        Properties  prop;
        WebDriver driver;

        @BeforeMethod
        public void setup() throws Exception {
                readData();
                driver = new ChromeDriver();

                driver.get(prop.getProperty("url"));
                driver.manage().window().maximize();
                driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(20));

                driver.findElement(By.xpath("//span[text()='My Account']")).click();
                driver.findElement(By.linkText("Login")).click();
        }

        @AfterMethod
        public void quitBrowser() throws Exception {
                Thread.sleep(3000);
                driver.quit();
        }
@DataProvider(name="supplyingTestData")
	public Object[][] supplyData() {

		Object[][] data = ReadExcel.getTestDataFromExcel("LoginData");
		return data;
	}

@Test(dataProvider = "supplyingTestData")
	public void loginCLassWork(String email, String password) throws Exception {

		driver.findElement(By.id("input-email")).sendKeys(email);
		driver.findElement(By.id("input-password")).sendKeys(password);
		driver.findElement(By.xpath("//input[@type='submit']")).click();

		String actualText = driver.findElement(By.linkText("Edit your account
information")).getText();
		String expectedText = "Edit your account information";

		//Hard Assert
		Assert.assertEquals(actualText, expectedText);

		System.out.println("Login success using username
:"+prop.getProperty("email")+" and Password :"+prop.getProperty("password"));
	}



	public void readData() throws IOException {
		String path = "D:\\Java-Selenium
Course\\Automation\\SeleniumCourse\\testData\\TestData.properties";
		FileInputStream fis = new FileInputStream(path) ;

		prop = new Properties();
		prop.load(fis);

		System.out.println(prop.getProperty("url"));

	}
}
```

Selenium Maven:

Maven is a powerful *project management tool* that is based on POM (project object model). It is used for projects build, dependency and documentation.

Maven has new features like dependency, which is used to download the dependency jar from the internet before the test execution.

With the help of Maven, we can execute the test scripts in the command line without an eclipse. And it always provides the framework folder structure.

Problems without Maven:

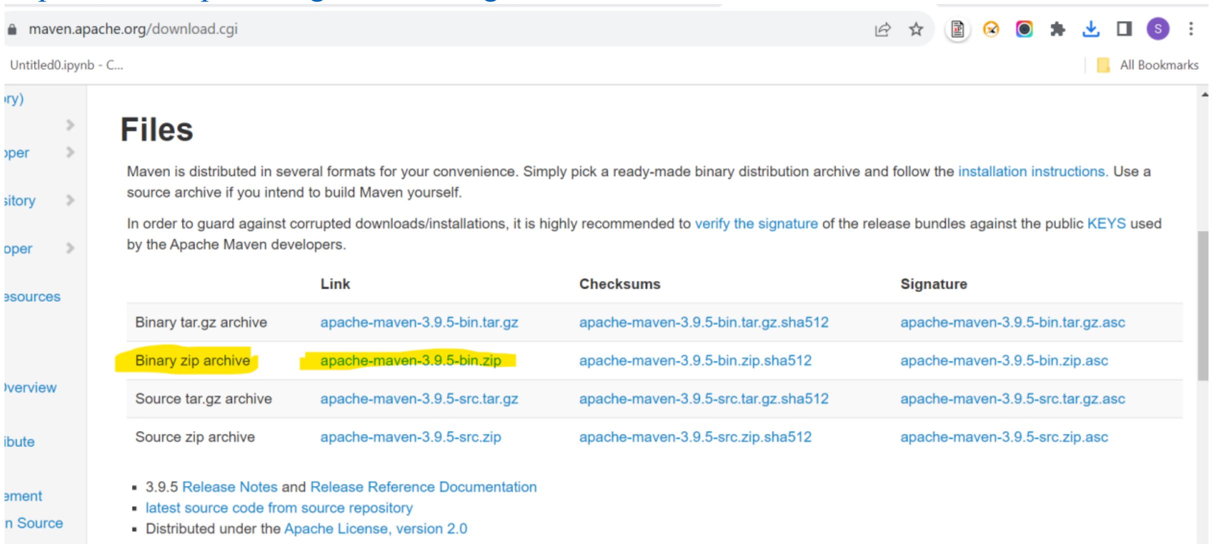
1. Adding set of Jars in each project
2. Creating the right project structure
3. **3.** Building and deploying the project

Maven resolves the above-mentioned problems and makes a project easy to build.

How to Install Maven

To install maven on windows, perform the following steps:

1. Download maven and extract it.
   Download latest Maven software (apache-maven-3.9.5-bin.zip) from site:
   https://maven.apache.org/download.cgi



maven.apache.org/download.cgi

Untitled0.ipynb - C...

All Bookmarks

ry)

per

sitory

oper

esources

verview

ibute

ement

n Source

Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the installation instructions. Use a source archive if you intend to build Maven yourself.

In order to guard against corrupted downloads/installations, it is highly recommended to verify the signature of the release bundles against the public KEYS used by the Apache Maven developers.

	Link	Checksums	Signature
Binary tar.gz archive	apache-maven-3.9.5-bin.tar.gz	apache-maven-3.9.5-bin.tar.gz.sha512	apache-maven-3.9.5-bin.tar.gz.asc
Binary zip archive	apache-maven-3.9.5-bin.zip	apache-maven-3.9.5-bin.zip.sha512	apache-maven-3.9.5-bin.zip.asc
Source tar.gz archive	apache-maven-3.9.5-src.tar.gz	apache-maven-3.9.5-src.tar.gz.sha512	apache-maven-3.9.5-src.tar.gz.asc
Source zip archive	apache-maven-3.9.5-src.zip	apache-maven-3.9.5-src.zip.sha512	apache-maven-3.9.5-src.zip.asc

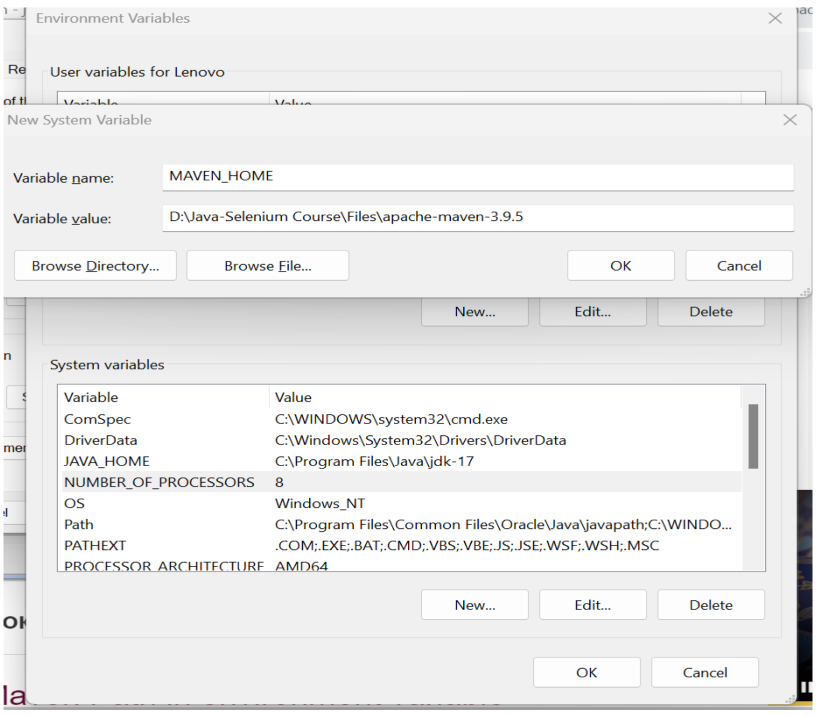
- 3.9.5 Release Notes and Release Reference Documentation
- latest source code from source repository
- Distributed under the Apache License, version 2.0

   Extract the files.

2. Add MAVEN_HOME in environment variable

Right click on **MyComputer** -> **properties** -> **Advanced System Settings** -> **Environment variables** -> **click new button**

Now **add MAVEN_HOME** in variable name and x. It must be the home directory of maven i.e. outer directory of bin. For example: **D:\Java-Selenium Course\apache-maven-3.9.5**.



Variable name: MAVEN_HOME
Variable value: D:\Java-Selenium Course\Files\apache-maven-3.9.5

Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
JAVA_HOME	C:\Program Files\Java\jdk-17
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\Program Files\Common Files\Oracle\Java\javapath;C:\WINDO...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64

Click on Ok.

3. Add maven path in environment variable

Now we are going to add the path of maven. Edit the existing **path** and append the path of maven. The path of maven should be **%MAVEN_HOME%/bin.** Example: **D:\Java-Selenium Course\apache-maven-3.9.5\bin**

Click on OK.

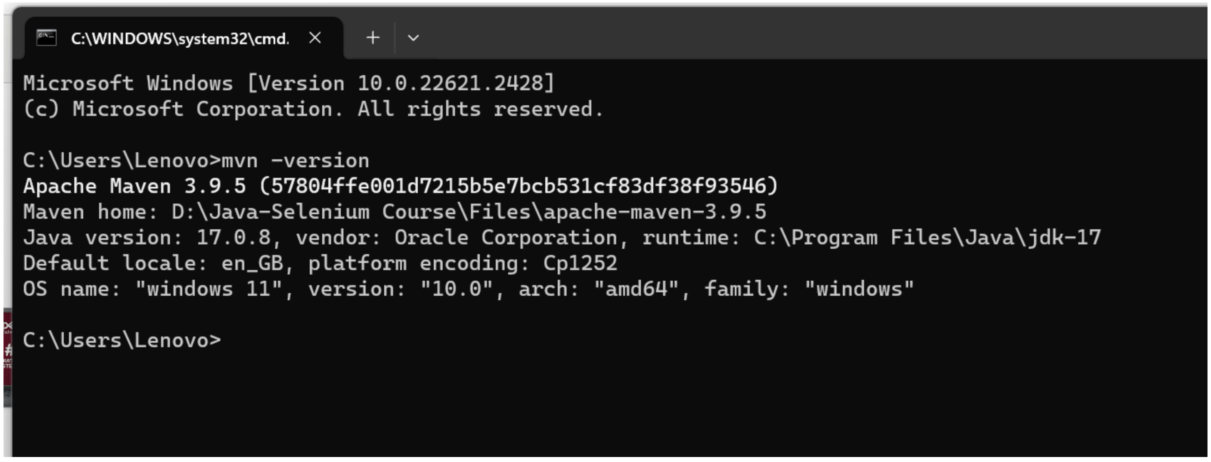
4. Verify Maven

To verify whether maven is installed or not, open the command prompt and write:

mvn –version

Now it will display the version of maven and jdk including the maven home and java home.

Looks like this:



```
Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>mvn -version
Apache Maven 3.9.5 (57804ffe001d7215b5e7bcb531cf83df38f93546)
Maven home: D:\Java-Selenium Course\Files\apache-maven-3.9.5
Java version: 17.0.8, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-17
Default locale: en_GB, platform encoding: Cp1252
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"

C:\Users\Lenovo>
```

Maven pom.xml file

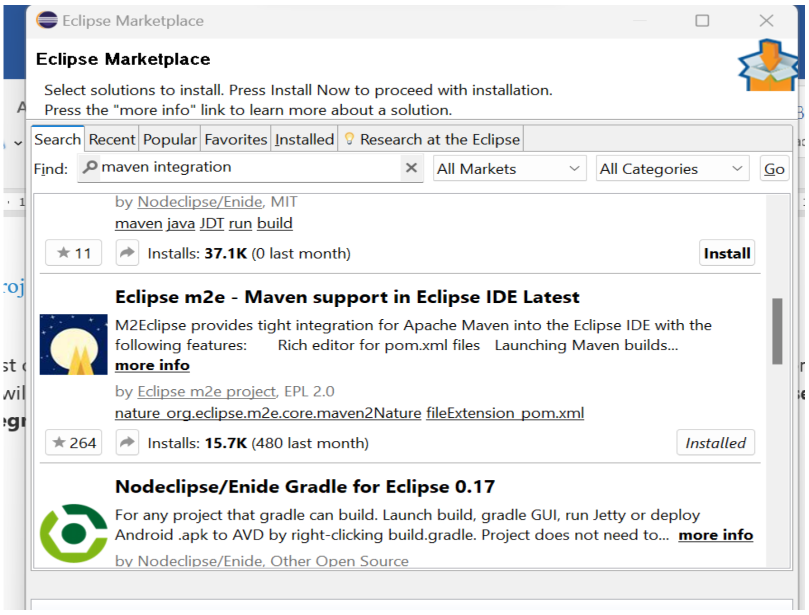
POM is an acronym for **Project Object Model**. The pom.xml file contains information of project and configuration information for the maven to build the project such as dependencies, build directory, source directory, test source directory, plugin, goals etc.

Maven reads the pom.xml file, then executes the goal.

Sample pom.xml file:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>selenium</groupId>
        <artifactId>selenium</artifactId>
        <version>0.0.1-SNAPSHOT</version>

        <dependencies>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <version>4.8.2</version>
                        <scope>test</scope>
                </dependency>
        </dependencies>
</project>
```

Most of the time, Maven plug-in is automatically installed in the Eclipse, but if it is not present, we will go to the **Eclipse Market Place** and search for **Maven** and download the **Eclipse M2E integrated version** from there.



Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.

Find: maven integration

by Nodeclipse/Enide, MIT
maven java JDT run build
★ 11 Installs: 37.1K (0 last month)

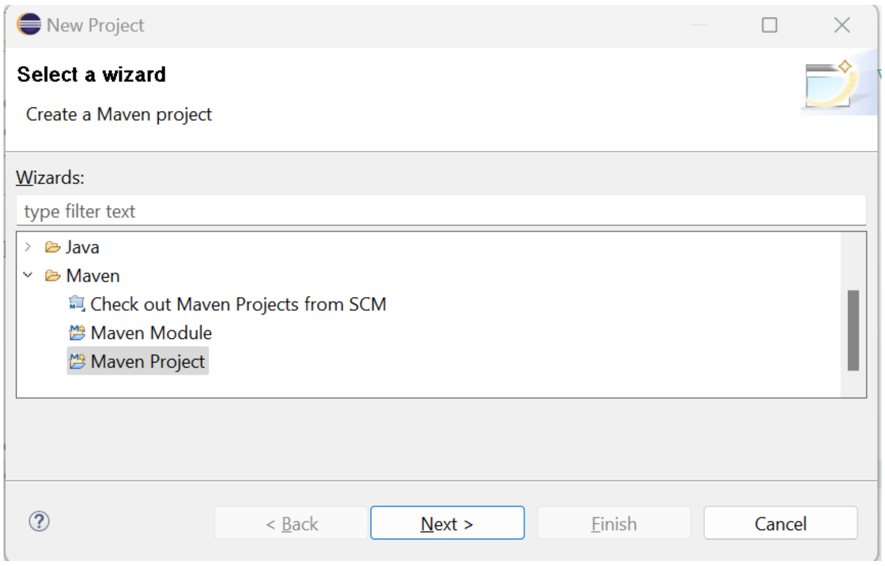
Eclipse m2e - Maven support in Eclipse IDE Latest
M2Eclipse provides tight integration for Apache Maven into the Eclipse IDE with the following features: Rich editor for pom.xml files Launching Maven builds... more info
by Eclipse m2e project, EPL 2.0
nature org.eclipse.m2e.core.maven2Nature fileExtension pom.xml
★ 264 Installs: 15.7K (480 last month)

Nodeclipse/Enide Gradle for Eclipse 0.17
For any project that gradle can build. Launch build, gradle GUI, run Jetty or deploy Android .apk to AVD by right-clicking build.gradle. Project does not need to... more info
by Nodeclipse/Enide, Other Open Source

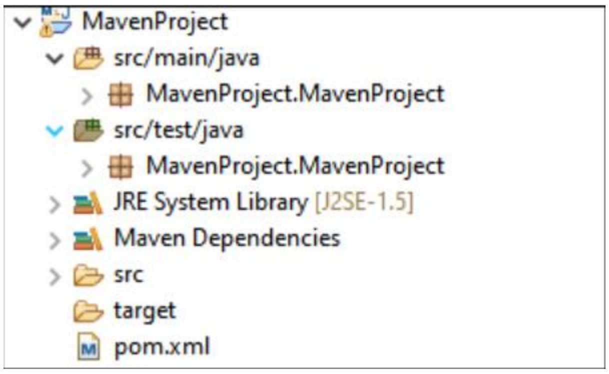
Create Maven project:

To create a Maven project, follow the below steps:

1. Go to the **File → New → Project**
2. Expand the **Maven** folder and select the **Maven Project** options, and click on **Next and Next** again without selecting anything.



3. Now select org.apache.maven.archetypes >> maven-archetype-quickstart and click on **Next.**
4. Provide the Group id as **MavenProject** and Artifact ID as **MavenProject**, and click on **Finish.**
5. Once we are done with creating the Maven project, our **Maven folder structure** will look like this:


MavenProject
src/main/java
MavenProject.MavenProject
src/test/java
MavenProject.MavenProject
JRE System Library [J2SE-1.5]
Maven Dependencies
src
target
pom.xml

We have created a Maven Project **“SeleniumClassAutomationProject”** and written our First login test as below:

```
package com.selenium.loginTest;

import java.time.Duration;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;



public class Login {

	WebDriver driver;

	@BeforeMethod
	public void startUp() {
		driver = new ChromeDriver();
		driver.get("https://tutorialsninja.com/demo/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(20));
		driver.findElement(By.xpath("//span[text()='My Account']")).click();
		driver.findElement(By.linkText("Login")).click();

	}

	@Test


	public void login() {

		driver.findElement(By.id("input-
email")).sendKeys("subbutop5@gmail.com");
		driver.findElement(By.id("input-password")).sendKeys("Password@123");
		driver.findElement(By.xpath("//input[@value='Login']")).click();

		String expectedText = "Edit your account information";
		String actualText = driver.findElement(By.linkText("Edit your account
information")).getText();
		Assert.assertEquals(expectedText, actualText, "Edit account link is not
prosent on thr page");

	}

	@AfterMethod
```

```
        public void quitBrowser() {
                driver.quit();
        }
}
```

TestNG.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite">
        <test thread-count="5" name="Test">
                <classes>
                        <class name="com.selenium.loginTest.Login">
                                <methods><include name="login"></include></methods>
                        </class>
                        <class name="com.selenium.loginTest.Login2" />
                </classes>
        </test> <!--
        Test -->
</suite> <!--
Suite -->
```

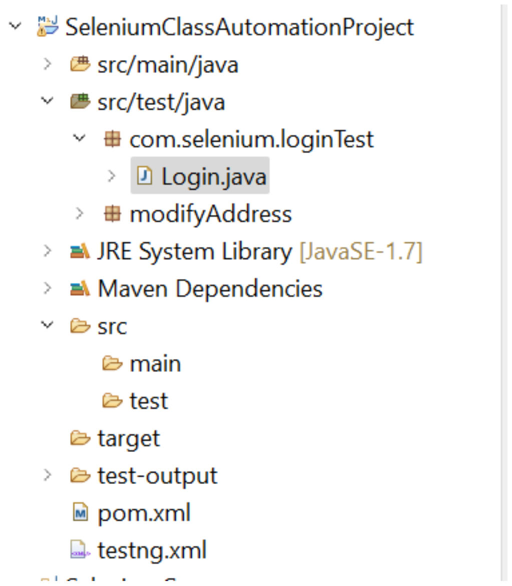
Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>Selenium</groupId>
        <artifactId>SeleniumClassAutomationProject</artifactId>
        <version>0.0.1-SNAPSHOT</version>

        <name>SeleniumClassAutomationProject</name>
        <!-- FIXME change it to the project's website -->
        <url>http://www.example.com</url>

        <properties>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
                <maven.compiler.source>1.7</maven.compiler.source>
                <maven.compiler.target>1.7</maven.compiler.target>
        </properties>

        <dependencies>
                <!--
```

```xml
https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
<dependency>
	<groupId>org.seleniumhq.selenium</groupId>
	<artifactId>selenium-java</artifactId>
	<version>4.15.0</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->
<dependency>
	<groupId>org.testng</groupId>
	<artifactId>testng</artifactId>
	<version>7.8.0</version>
	<scope>test</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->
<dependency>
	<groupId>org.apache.poi</groupId>
	<artifactId>poi</artifactId>
	<version>5.2.4</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->
<dependency>
	<groupId>org.apache.poi</groupId>
	<artifactId>poi-ooxml</artifactId>
	<version>5.2.4</version>
</dependency>


<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml-schemas -->
<dependency>
	<groupId>org.apache.poi</groupId>
	<artifactId>poi-ooxml-schemas</artifactId>
	<version>4.1.2</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-scratchpad -->
<dependency>
	<groupId>org.apache.poi</groupId>
	<artifactId>poi-scratchpad</artifactId>
	<version>5.2.4</version>
</dependency>

<!-- https://mvnrepository.com/artifact/commons-io/commons-io -->
<dependency>
	<groupId>commons-io</groupId>
	<artifactId>commons-io</artifactId>
	<version>2.15.0</version>
</dependency>
```

```
        </dependencies>
        </build>
</project>
```

Writing First Test Scenario and Test Cases:



```
SeleniumClassAutomationProject
  src/main/java
  src/test/java
    com.selenium.loginTest
      Login.java
    modifyAddress
  JRE System Library [JavaSE-1.7]
  Maven Dependencies
  src
    main
    test
  target
  test-output
  pom.xml
  testng.xml
```

In the above Screenshot we have Created Login.java file and written Four test cases for Login scenario. Refer the code below:

```
public class Login {

        WebDriver driver;

        @BeforeMethod
        public void startUp() {
                driver = new ChromeDriver();
                driver.get("https://tutorialsninja.com/demo/");
                driver.manage().window().maximize();
                driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(20));
                driver.findElement(By.xpath("//span[text()='My Account']")).click();
                driver.findElement(By.linkText("Login")).click();
        }
        @Test

        public void loginWithValidCredentials() {
                driver.findElement(By.id("input-
email")).sendKeys("subbutop5@gmail.com");
                driver.findElement(By.id("input-password")).sendKeys("Password@123");
                driver.findElement(By.xpath("//input[@value='Login']")).click();
                String expectedText = "Edit your account information";
```

```
		String actualText = driver.findElement(By.linkText("Edit your account 
information")).getText();
		Assert.assertEquals(expectedText, actualText, "Edit account link is not 
prosent on thr page");
	}

	@Test
	public void loginWithValidUsernameAndWrongPassword() {
		driver.findElement(By.id("input-
email")).sendKeys("subbutop5@gmail.com");
		driver.findElement(By.id("input-password")).sendKeys("1234jsdjasdasd");
		driver.findElement(By.xpath("//input[@value='Login']")).click();
		String loginErrorText = driver.findElement(By.xpath("//div[text()='Warning: 
No match for E-Mail Address and/or Password.']")).getText();
		Assert.assertTrue(loginErrorText.contains("No match"), "Login Error 
message is not getting displayed");
	}

	@Test
	public void loginWithInvalidUsernameAndCPassword() {
		driver.findElement(By.id("input-
email")).sendKeys("123456789@gmail.com");
		driver.findElement(By.id("input-password")).sendKeys("Password@123");
		driver.findElement(By.xpath("//input[@value='Login']")).click();
		String loginErrorText = driver.findElement(By.xpath("//div[text()='Warning: 
No match for E-Mail Address and/or Password.']")).getText();
		Assert.assertTrue(loginErrorText.contains("No match"), "Login Error 
message is not getting displayed");
	}

	@Test
	public void loginWithNoCredentials() {
		driver.findElement(By.id("input-email")).sendKeys("");
		driver.findElement(By.id("input-password")).sendKeys("");
		driver.findElement(By.xpath("//input[@value='Login']")).click();
		String loginErrorText = driver.findElement(By.xpath("//div[text()='Warning: 
No match for E-Mail Address and/or Password.']")).getText();
		Assert.assertTrue(loginErrorText.contains("No match for E-Mail Address 
and/or Password"), "Login Error message is not getting displayed");
	}

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

We have Created a ‘Register.java’ class and written Three test cases for Create Account scenario. Fins the code below:

```
public class Register {
	WebDriver driver;

	@BeforeMethod
	public void startUp() {
		String browserName = "chrome";
		if (browserName.equals("chrome")) {
			driver = new ChromeDriver();
		} else if (browserName.equals("firefox")) {
			driver = new FirefoxDriver();
		} else if (browserName.equals("edge")) {
			driver = new EdgeDriver();
		} else {
			System.out.println("No Browser Found!!");
		}
		driver.get("https://tutorialsninja.com/demo/");
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
		driver.findElement(By.xpath("//span[text()='My Account']")).click();
		driver.findElement(By.linkText("Register")).click();
	}

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

	@Test(priority = 1)
	public void registerAccountWithValidFields() throws Exception {

		Random rnd = new Random();
	   int number = rnd.nextInt(99999);

		driver.findElement(By.id("input-firstname")).sendKeys("Subhash");
		driver.findElement(By.id("input-lastname")).sendKeys("Sunagar");;
		driver.findElement(By.id("input-
email")).sendKeys("subhash"+number+"@gmail.com");
		driver.findElement(By.name("telephone")).sendKeys("1234567890");
		driver.findElement(By.id("input-password")).sendKeys("Pass@123");
		driver.findElement(By.id("input-confirm")).sendKeys("Pass@123");
		driver.findElement(By.xpath("//input[@name='agree']")).click();
		driver.findElement(By.xpath("//input[@value='Continue']")).click();


		String accountSuccessMessage =
driver.findElement(By.xpath("//h1[contains(text(),'Account Has Been Created')]")).getText();
```

```
                Assert.assertTrue(accountSuccessMessage.contains("Account Has Been
Created"), "Account has not been created!!!");
        }

        @Test(priority = 2)
        public void registerAccountWithValidFieldsAndSubscribeAsYes() {
                Random rnd = new Random();
           int number = rnd.nextInt(99999);

                driver.findElement(By.id("input-firstname")).sendKeys("Subhash");
                driver.findElement(By.id("input-lastname")).sendKeys("Sunagar");
                driver.findElement(By.id("input-
email")).sendKeys("subhash"+number+"@gmail.com");
                driver.findElement(By.name("telephone")).sendKeys("1234567890");
                driver.findElement(By.id("input-password")).sendKeys("Pass@123");
                driver.findElement(By.id("input-confirm")).sendKeys("Pass@123");
                driver.findElement(By.xpath("//input[@name='newsletter' and
@value='1']")).click();
                driver.findElement(By.xpath("//input[@name='agree']")).click();
                driver.findElement(By.xpath("//input[@value='Continue']")).click();

                String accountSuccessMessage =
driver.findElement(By.xpath("//h1[contains(text(),'Account Has Been Created')]")).getText();
                Assert.assertTrue(accountSuccessMessage.contains("Account Has Been
Created"), "Account has not been created!!!");
        }

        @Test(priority = 3)
        public void registerAccountWithoutAgreeingToPrivacyPolicy() throws IOException
        {
                Random rnd = new Random();
           int number = rnd.nextInt(99999);

                driver.findElement(By.id("input-firstname")).sendKeys("Subhash");
                driver.findElement(By.id("input-lastname")).sendKeys("Sunagar");
                driver.findElement(By.id("input-
email")).sendKeys("subhash"+number+"@gmail.com");
                driver.findElement(By.name("telephone")).sendKeys("1234567890");
                driver.findElement(By.id("input-password")).sendKeys("Pass@123");
                driver.findElement(By.id("input-confirm")).sendKeys("Pass@123");
                driver.findElement(By.xpath("//input[@name='newsletter' and
@value='1']")).click();
                //Here we are not agreeing the Privacy Policy
                driver.findElement(By.xpath("//input[@value='Continue']")).click();

                String actualWarning =
driver.findElement(By.xpath("//div[contains(@class,'alert-dismissible')]")).getText();
                String expectedWarning = "Warning: You must agree to the Privacy Policy!";
                Assert.assertEquals(actualWarning, expectedWarning, "Privacy Policy error
message did not get displayed");
```

```
		TakesScreenshot screenshot = (TakesScreenshot) driver;
		File src = screenshot.getScreenshotAs(OutputType.FILE);
		File dst = new File("D:\\Java-Selenium 
Course\\Automation\\SeleniumClassAutomationProject\\screenshots\\RegisterPageScreensho
t.png");
		FileUtils.copyFile(src,dst);
	}


	//Home-Work
	@Test
	public void registerAccountWithoutEnteringAnyFields() {

	}
}
```

Taking Screenshot in Selenium:

```
public class TakeScreenshot {

	WebDriver driver;
	public void takingScreenshot() throws IOException {

		driver = new ChromeDriver();
		driver.get("https://www.google.com");
		//Convert web driver object to TakesScreenshot
		TakesScreenshot screenShot = (TakesScreenshot) driver;
		//Call getScreenshotAs method to create image file
		File source = screenShot.getScreenshotAs(OutputType.FILE);
		//Move image file to new destination
		File dest = new File("D:\\Java-Selenium 
Course\\Automation\\SeleniumClassAutomationProject\\screenshots\\SeleniumTesting.png");
		//Copy file at destination
		FileUtils.copyFile(source, dest);
		driver.quit();
	}
```

Framework code with Base Class and Login Class:

Simplify Reading data from config.properties file in our framework written in Base class:

1. **First way is** : Reading data from config.properties file by calling the **readConfigFileData()** method in every Class @BeforeMethod so as to use **prop** data.

As shown in below code:

Base Class code is given below:

```
public class Base{

	WebDriver driver;
	public Properties prop;

	public void readConfigFileData() {
		String path = 
System.getProperty("user.dir")+"\\src\\main\\java\\com\\selenium\\config\\config.properties";
		prop = new Properties();
		try {
			FileInputStream fis = new FileInputStream(path);
			prop.load(fis);
		} catch (Throwable e) {
			e.printStackTrace();
		}
		System.out.println("url is : "+prop.getProperty("url"));
	}

	public WebDriver initializeBrowserAndLaunchApplicationURL(String 
browserName){

		if (browserName.equals("chrome")) {
			driver = new ChromeDriver();
		} else if (browserName.equals("firefox")) {
			driver = new FirefoxDriver();
		} else if (browserName.equals("edge")) {
			driver = new EdgeDriver();
		} else {
			System.out.println("No Browser Found!!");
		}
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
		driver.get(prop.getProperty("url"));
		return driver;
	}
}
```

```
public class Login extends Base {
	WebDriver driver;

	@BeforeMethod
	public void startUp(){
		readConfigFileData();
		driver = 
initializeBrowserAndLaunchApplicationURL(prop.getProperty("browserName"));
		driver.findElement(By.xpath("//span[text()='My Account']")).click();
		driver.findElement(By.linkText("Login")).click();
	}

	@Test(priority = 1)

	public void loginWithValidCredentials() {
		driver.findElement(By.id("input-
email")).sendKeys("subbutop5@gmail.com");
		driver.findElement(By.id("input-password")).sendKeys("Password@123");
		driver.findElement(By.xpath("//input[@value='Login']")).click();
		String expectedText = "Edit your account information";
		String actualText = driver.findElement(By.linkText("Edit your account 
information")).getText();
		Assert.assertEquals(expectedText, actualText, "Edit account link is not 
prosent on thr page");
	}
}
```

2. **Second way is:** Reading data from config.properties file by writing the code (read data from config file code) in a Base Class Constructor. And in Login class, create Login class Constructor and call the parent class (Base class) constructor using the **super().** Code is as shown below:

Base.java

```
public class Base {

	WebDriver driver;
	public Properties prop;

	public Base(){
		String path = 
System.getProperty("user.dir")+"\\src\\main\\java\\com\\selenium\\config\\config.properties";
		prop = new Properties();
		try {
			FileInputStream fis = new FileInputStream(path);
			prop.load(fis);
		} catch (Throwable e) {
			e.printStackTrace();
		}
		System.out.println("url is : "+prop.getProperty("url"));
	}

	public WebDriver initializeBrowserAndLaunchApplicationURL(String 
browserName){

		if (browserName.equals("chrome")) {
			driver = new ChromeDriver();
		} else if (browserName.equals("firefox")) {
			driver = new FirefoxDriver();
		} else if (browserName.equals("edge")) {
			driver = new EdgeDriver();
		} else {
			System.out.println("No Browser Found!!");
		}
		driver.manage().window().maximize();
		driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
		driver.get(prop.getProperty("url"));
		return driver;
	}
}
```

Login.java:

```
public class Login extends Base {

	WebDriver driver;

	public Login() {
		super();
	}

	@BeforeMethod
	public void startUp(){

		driver = 
initializeBrowserAndLaunchApplicationURL(prop.getProperty("browserName"));
		driver.findElement(By.xpath("//span[text()='My Account']")).click();
		driver.findElement(By.linkText("Login")).click();
	}

	@Test(priority = 1)
	public void loginWithValidCredentials() {
		driver.findElement(By.id("input-
email")).sendKeys("subbutop5@gmail.com");
		driver.findElement(By.id("input-password")).sendKeys("Password@123");
		driver.findElement(By.xpath("//input[@value='Login']")).click();
		String expectedText = "Edit your account information";
		String actualText = driver.findElement(By.linkText("Edit your account 
information")).getText();
```

```
                    Assert.assertEquals(expectedText, actualText, "Edit account link is not
prosent on thr page");
          }
}
```

Hybrid TestNG framework using POM - Page Object Model Design Pattern and PageFactory.

--Here all the logic steps written in test classes only, if any particular locator/webElement changes, then we need to go to individual test methods and change them.

--Thats a difficult task when we have 1000s of test cases, hence we need to centralize these locators and logic to some other place so that these can be modified from one place.

Advantages of doing above step:

1. Readability of the code
2. Re-usability of the code
3. Maintenance of the code

--Integrating Extent Report and attaching Screenshots - in selenium Framework.

This requires implementing Listeners class and ExtentReporter class

- Implement Listeners class (Belongs to TestNG)

- to print statements/logs in console or in extent report, we need to update testNG.xml file with below code:

```
  <listeners>
     <listener class-name="com.selenium.listeners.MyListeners"/>
  </listeners>
```

- https://www.extentreports.com/ - we have different type of reporters

- ExtentSparkReporter and many more

--Attach Screenshot to Extent Report

--Auto Launch Extent Report after execution

Taking Screenshot – In Utilities Class

```
public static String takeScreenshot(WebDriver driver, String screenshotName) throws
Throwable {
        TakesScreenshot screenshot = (TakesScreenshot) driver;
        File src = screenshot.getScreenshotAs(OutputType.FILE);
        String dstScreenshotPath = "D:\\Java-Selenium
Course\\Automation\\SeleniumCaseStudyAutomationProject\\screenshots\\"+screenshotNam
e+"Screenshot.png";
        FileUtils.copyFile(src,new File(dstScreenshotPath));
        System.out.println("Screenshot has been taken");
        return dstScreenshotPath;
}
```

Call the method from Utilities class to take Screenshot and attach it to Extent report in MyListeners class using the below code:

```
destScreenshotPath = Utilities.takeScreenshot(driver,
result.getName());
```

```
extentTest.addScreenCaptureFromPath(destScreenshotPath);
```

Auto Launch Extent Report

```
//Below code is to open the extent report Automatically after the execution of all the test
cases is over.
String path= System.getProperty("user.dir")+"\\test-output\\ExtentReport\\extentReport.html";
File extentReport = new File(path);
        try {
                Desktop.getDesktop().browse(extentReport.toURI());
        } catch (IOException e) {
                e.printStackTrace();
}
```

MyListener.java – code

```
public class MyListeners implements ITestListener {
        ExtentReports extentReport;
        ExtentTest extentTest;
        WebDriver driver;
        String destScreenshotPath;

        @Override
        public void onStart(ITestContext context) {
```

```
		extentReport = ExtentReporter.generateExtentReport();

	}

	@Override
	public void onTestStart(ITestResult result) {
		extentTest = extentReport.createTest(result.getName());
		extentTest.log(Status.INFO, result.getName() + " - started executing");
	}

	@Override
	public void onTestSuccess(ITestResult result) {
		extentTest.log(Status.PASS, result.getName() + " - got successfully executed");
	}

	@Override
	public void onTestFailure(ITestResult result) {
	  try {
		driver = (WebDriver) result.getTestClass().getRealClass().getDeclaredField("driver")
					.get(result.getInstance());
		} catch (IllegalArgumentException | IllegalAccessException | NoSuchFieldException |
SecurityException e) {
		e.printStackTrace();
	  }

	// take screenshot when test failed
	/*
	TakesScreenshot screenshot = (TakesScreenshot) driver;
	File srcScreenshot = screenshot.getScreenshotAs(OutputType.FILE);
	String destScreenshotPath =
System.getProperty("user.dir")+"\\screenshots\\"+result.getName()+".png";
	try {
		FileUtils.copyFile(srcScreenshot, new File(destScreenshotPath));
		//FileHandler.copy(srcScreenshot, new File(destScreenshotPath));
	} catch (IOException e) {
		e.printStackTrace();
	}
	  */

	try {
		//Calling the screenshot method from Utilities Class
		destScreenshotPath = Utilities.takeScreenshot(driver, result.getName());
	} catch (Throwable e) {
		e.printStackTrace();
	}
		extentTest.addScreenCaptureFromPath(destScreenshotPath);
		extentTest.log(Status.INFO, result.getThrowable());
		extentTest.log(Status.FAIL, result.getName() + " - got failed");
	}

	@Override
	public void onTestSkipped(ITestResult result) {
		extentTest.log(Status.INFO, result.getThrowable());
		extentTest.log(Status.SKIP, result.getName() + " - got skipped");

		System.out.println(result.getName() + " - got skipped");
		System.out.println(result.getThrowable());
	}

	@Override
	public void onFinish(ITestContext context) {
		System.out.println("Execution of Project Tests Finished");
		extentTest.log(Status.INFO, "Execution of Project Tests Finished");
		extentReport.flush();


		File path = new File(System.getProperty("user.dir")+"\\test-
		output\\ExtentReports\\extentReport.html");
		try {
			Desktop.getDesktop().browse(path.toURI());
		} catch (IOException e) {
			e.printStackTrace();
		}
	}
}
```

ExtentReporter.java code

```
public class ExtentReporter {

	public static ExtentReports generateExtentReport() {
	ExtentReports extentReport = new ExtentReports();

		File extentReportPath = new File(System.getProperty("user.dir") + "\\test-
		output\\ExtentReports\\extentReport.html");
		ExtentSparkReporter sparkReporter = new ExtentSparkReporter(extentReportPath);

		sparkReporter.config().setTheme(Theme.DARK);
		sparkReporter.config().setReportName("Selenium Project Report");
		sparkReporter.config().setDocumentTitle("Selenium Automation Report");
		sparkReporter.config().setTimeStampFormat("dd/MM/yyyy hh:mm:ss");

		extentReport.attachReporter(sparkReporter);

		extentReport.setSystemInfo("Application URL","https://tutorialsninja.com/demo/");
		extentReport.setSystemInfo("Browser Name", "chrome");
		extentReport.setSystemInfo("Operating System", System.getProperty("os.name"));
		extentReport.setSystemInfo("Java Version", System.getProperty("java.version"));

		return extentReport;
	}
}
```

Run the Automation scripts using Maven instead of TestNG.xml file

Because in Continuation Integration (Jenkins) we need to run the scripts using maven and will not rely on testNG.xml file.

Right click on Project >> Run As >> Maven Test.

- But none of the test cases will run if we do not have the **maven surefire plugin** dependency in pom.xml. So, add the surefire plugin in your pom.xml file.

```
<plugin>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.22.1</version>
</plugin>
```

- And we are instructing maven to run all the **classes** which have the **Test** keyword

- With these changes all the test cases ran successfully – **But** the extent report will not be generated and screenshots will not be taken because those are the part of Listeners. This is because testNG.xml is not involved in this current project run using Maven Test.

- Solution is to make this maven to invoke this pom.xml file and pom.xml should invoke the testing.xml file instead of invoking the classes having Test keyword.

- Update the surefire plugin with the below code in pom.xml

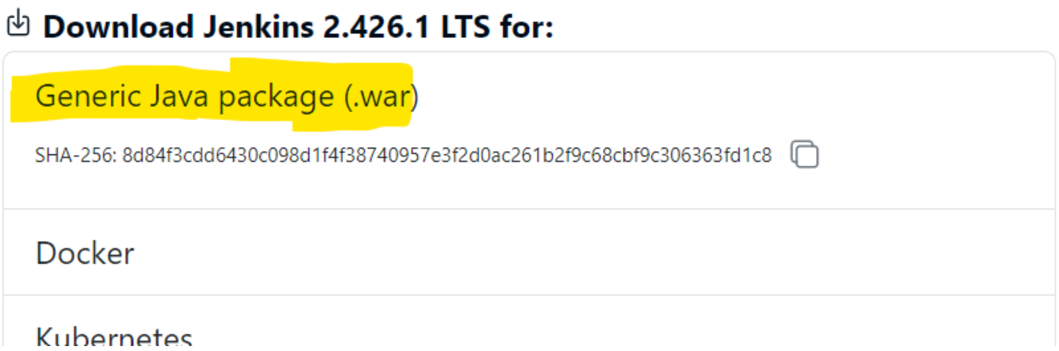
```
<plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.2.2</version>
        <configuration>
                <suiteXmlFiles>
                        <suiteXmlFile>testng.xml</suiteXmlFile>
                </suiteXmlFiles>
        </configuration>
</plugin>
```

Run Maven project in Command Prompt without Eclipse IDE.

To achieve this, we need to **setup Maven in our machine**
Open command prompt in a project location Enter the command >> **mvn test**
And all the test cases will run successfully without any IDEs.

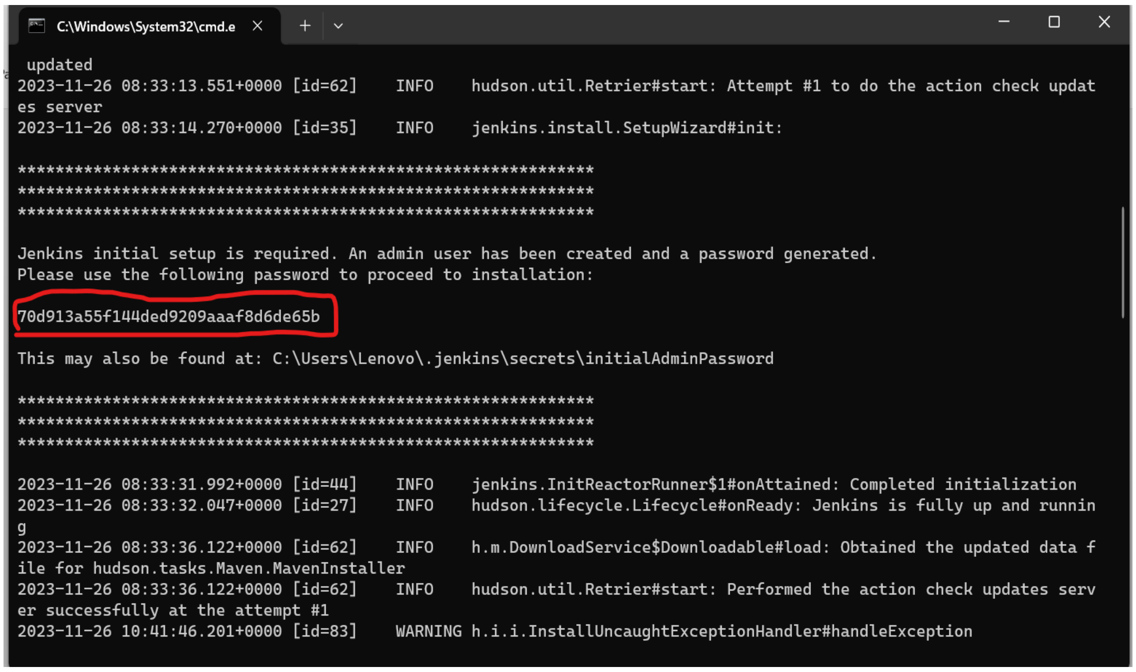
Jenkins Integration

- Download Jenkins from https://www.jenkins.io/download/
    - Download Generic Java Package (.war)


Download Jenkins 2.426.1 LTS for:
Generic Java package (.war)
SHA-256: 8d84f3cdd6430c098d1f4f38740957e3f2d0ac261b2f9c68cbf9c306363fd1c8
Docker
Kubernetes

- Place the war file in any folder of your machine and run the below command in Command Prompt (cmd)

  Java -jar jenkins.war



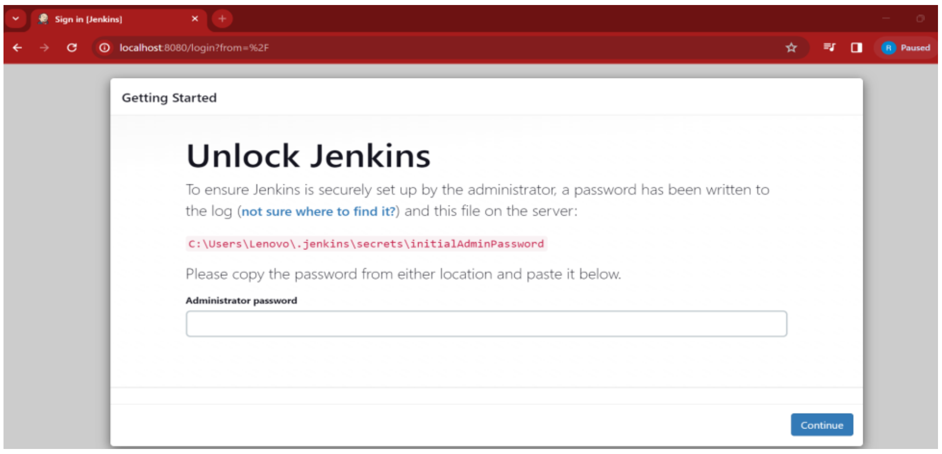
```
 updated
2023-11-26 08:33:13.551+0000 [id=62]    INFO    hudson.util.Retrier#start: Attempt #1 to do the action check updat
es server
2023-11-26 08:33:14.270+0000 [id=35]    INFO    jenkins.install.SetupWizard#init:

*************************************************************
*************************************************************
*************************************************************

Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

70d913a55f144ded9209aaaf8d6de65b

This may also be found at: C:\Users\Lenovo\.jenkins\secrets\initialAdminPassword

*************************************************************
*************************************************************
*************************************************************

2023-11-26 08:33:31.992+0000 [id=44]    INFO    jenkins.InitReactorRunner$1#onAttained: Completed initialization
2023-11-26 08:33:32.047+0000 [id=27]    INFO    hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and runnin
g
2023-11-26 08:33:36.122+0000 [id=62]    INFO    h.m.DownloadService$Downloadable#load: Obtained the updated data f
ile for hudson.tasks.Maven.MavenInstaller
2023-11-26 08:33:36.122+0000 [id=62]    INFO    hudson.util.Retrier#start: Performed the action check updates serv
er successfully at the attempt #1
2023-11-26 10:41:46.201+0000 [id=83]    WARNING h.i.i.InstallUncaughtExceptionHandler#handleException
```

- And copy the **admin password** from the command prompt or the password may also be found in the below location

  C:\Users\Lenovo\.jenkins\secrets\initialAdminPassword

- With this, Jenkins will automatically be installed in your machine and start running in port **8080**.
  - There may be some situations where in 8080 port other software's may also running in our machine then the conflict will come.
  - i.e., there is one slot 8080 port where another software other than Jenkins is running So Jenkins cannot apply and this may not be possible.
  - In this case we need to run the Jenkins in different port number using the command
    - java -jar Jenkins.war –httpPort=9090
  - Now open any browser and launch Jenkins with the url: http://localhost:9090/

- Open any browser and launch url: http://localhost:8080/ and below page will open



Unlock Jenkins

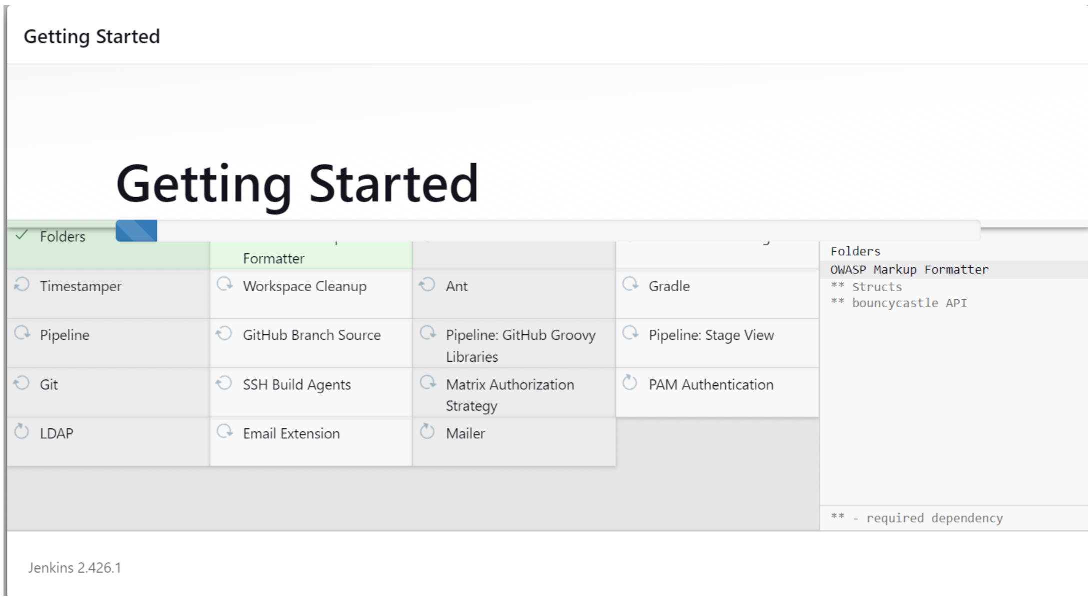
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

```
C:\Users\Lenovo\.jenkins\secrets\initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

Enter the password and click on Continue.
- Select "Install Suggested Plugins"
- Wait for all the plugins gets installed



```
Folders
OWASP Markup Formatter
** Structs
** bouncycastle API

** - required dependency
```

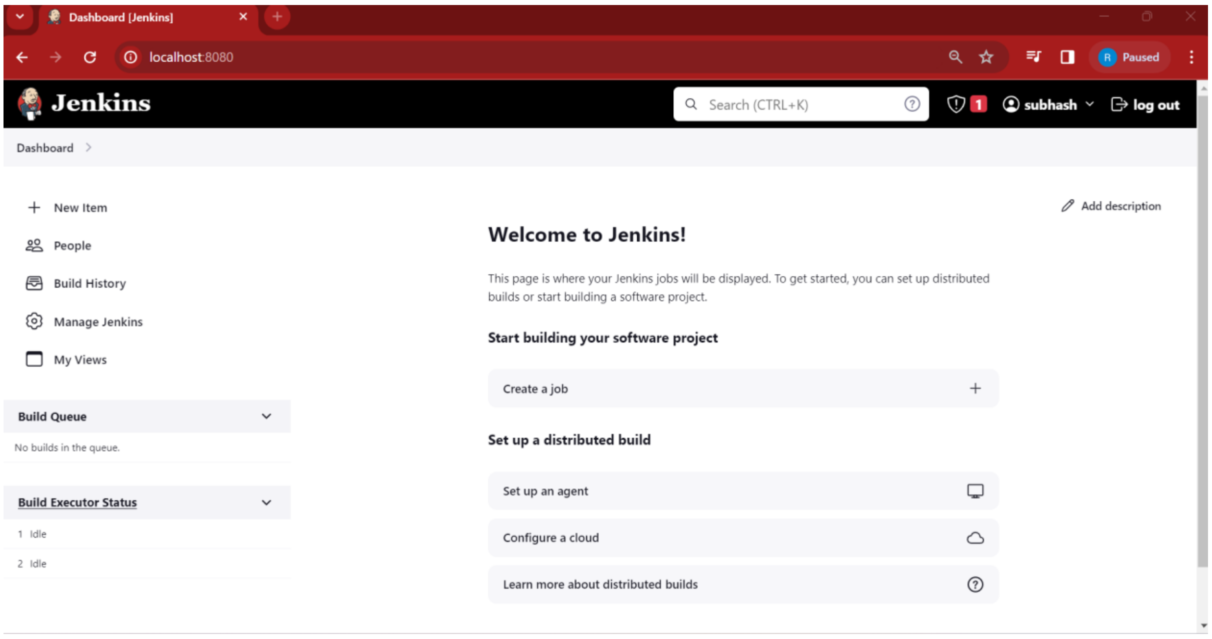
Once the installation is done, **Create Admin User and Password**

Username: subhash

Password: 12345

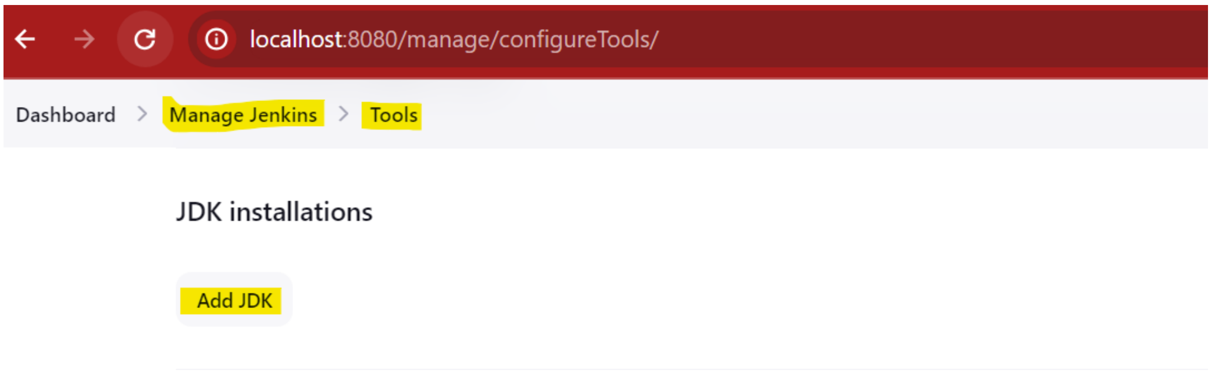
Jenkins Url: http://localhost:8080/

- Click on "Start Using Jenkins"
- Now our Jenkins is ready to use and the below Jenkins Dashboard page will open.

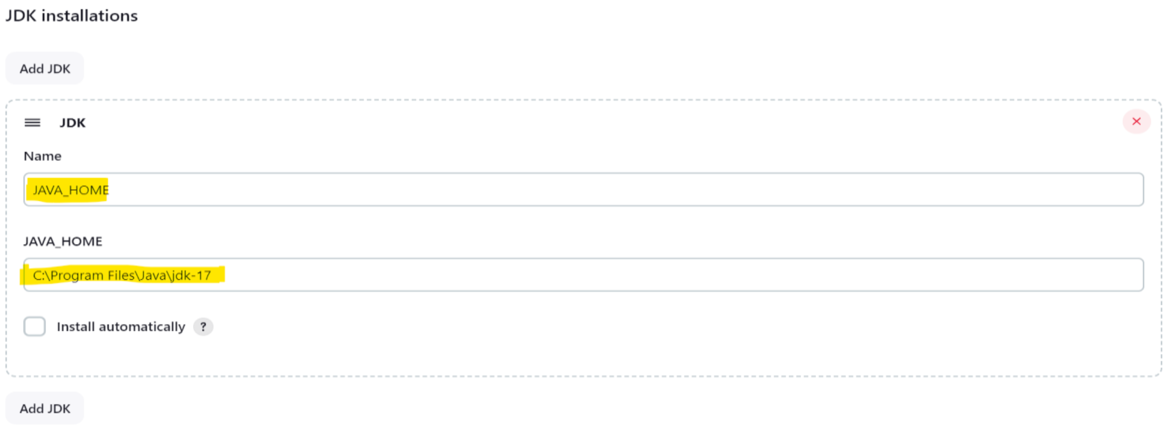


Now Config Jenkins:

1. **Add JDK :** Go to '**Manage Jenkins**' >> Tools >> Select JDK Option


localhost:8080/manage/configureTools/
Dashboard > Manage Jenkins > Tools
JDK installations
Add JDK

>> Provide **JAVA_HOME** path (which is installed in our machine)



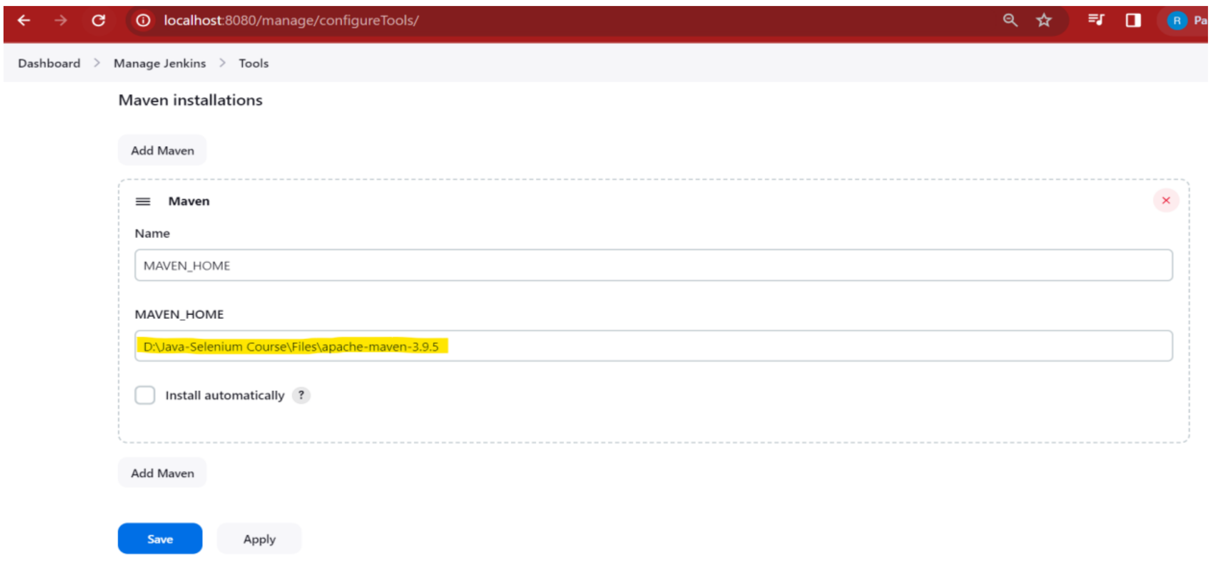
Name: JAVA_HOME

JAVA_HOME: C:\Program Files\Java\jdk-17

Install automatically: unchecked

2. Select **Maven Installation** >> Add Maven >> un-check 'install automatically' option

And add MAVEN_HOME path as shown below


localhost:8080/manage/configureTools/
Dashboard > Manage Jenkins > Tools
Add Maven
Add Maven
Save
Apply

Maven installations

Maven

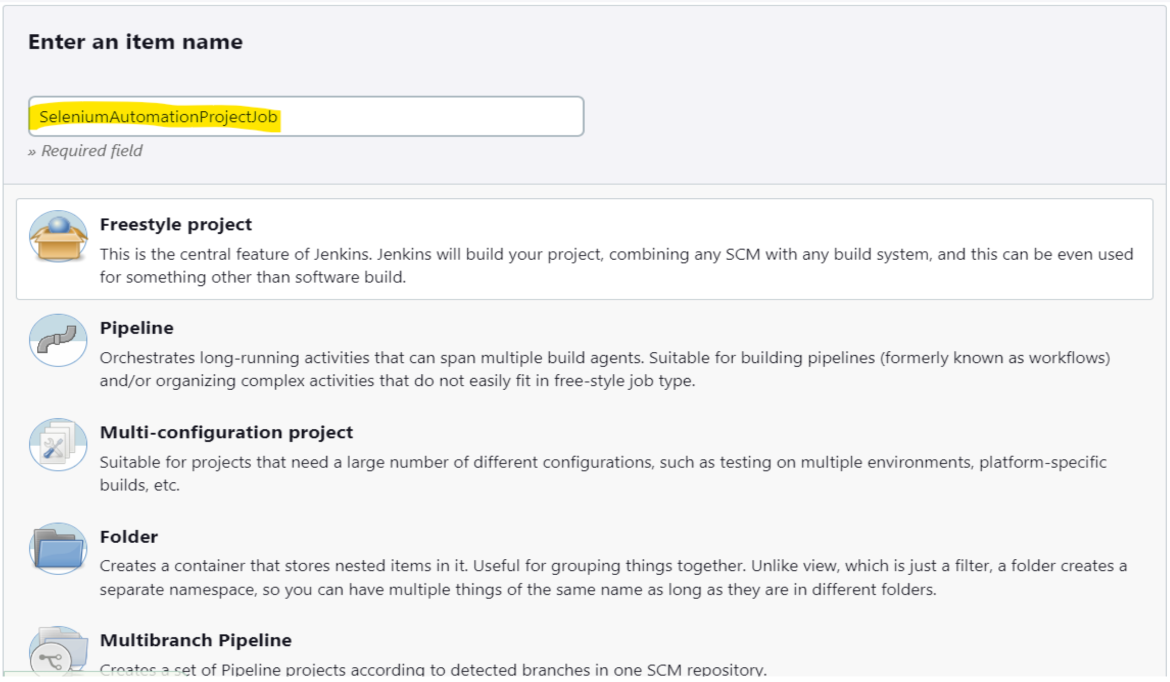
Name: MAVEN_HOME

MAVEN_HOME: D:\Java-Selenium Course\Files\apache-maven-3.9.5

☐ Install automatically ?

Click on Save.

3. Click on **'Create Job'** from dashboard page >> enter 'item name' and >> select 'Freestyle project' >> click on OK



Enter an item name

SeleniumAutomationProjectJob

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

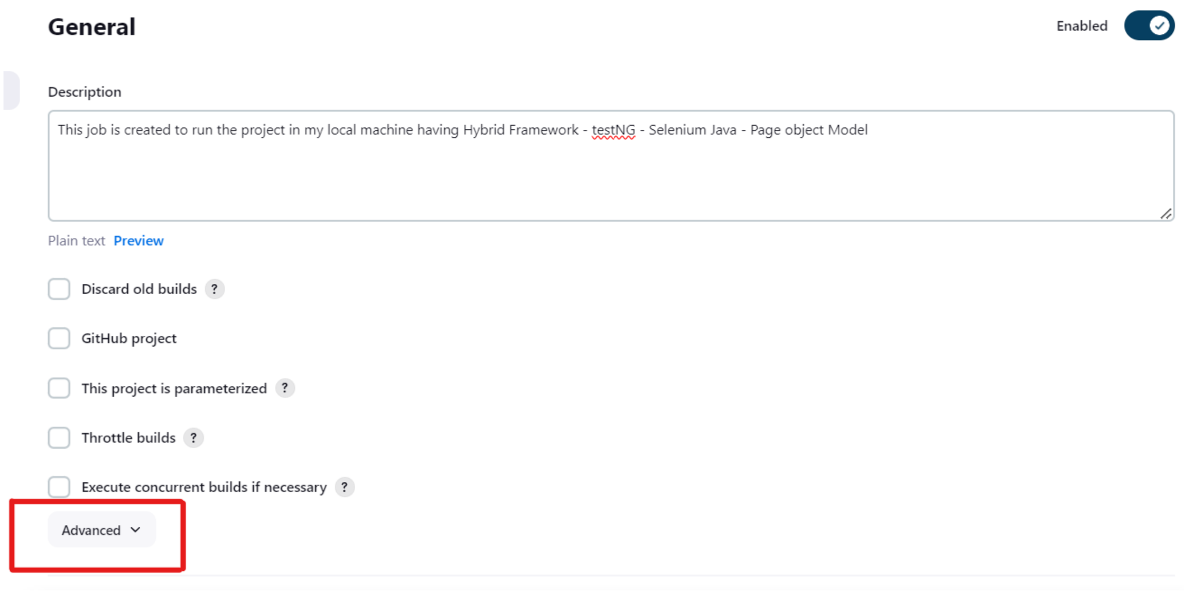
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

a. Now we need to configure the created 'SeleniumAutomationProjectJob' >> General >> click on 'Advanced' option


General

Enabled

Description

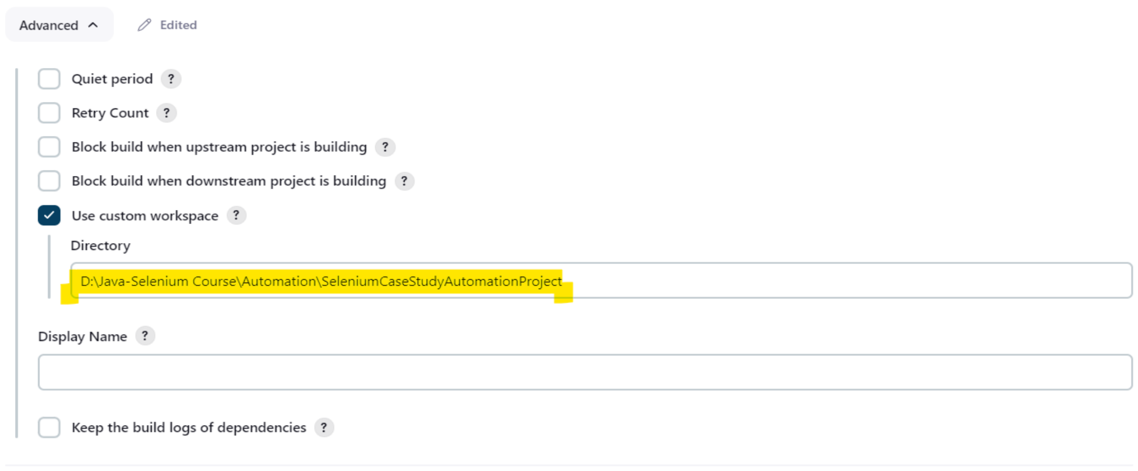
This job is created to run the project in my local machine having Hybrid Framework - testNG - Selenium Java - Page object Model

Plain text Preview

- [ ] Discard old builds
- [ ] GitHub project
- [ ] This project is parameterized
- [ ] Throttle builds
- [ ] Execute concurrent builds if necessary

Advanced

b. Now select 'custom workspace' checkbox >> Directory – provide the path of your project directory where your project **pom.xml** file is present


Advanced Edited

- [ ] Quiet period
- [ ] Retry Count
- [ ] Block build when upstream project is building
- [ ] Block build when downstream project is building
- [x] Use custom workspace

Directory

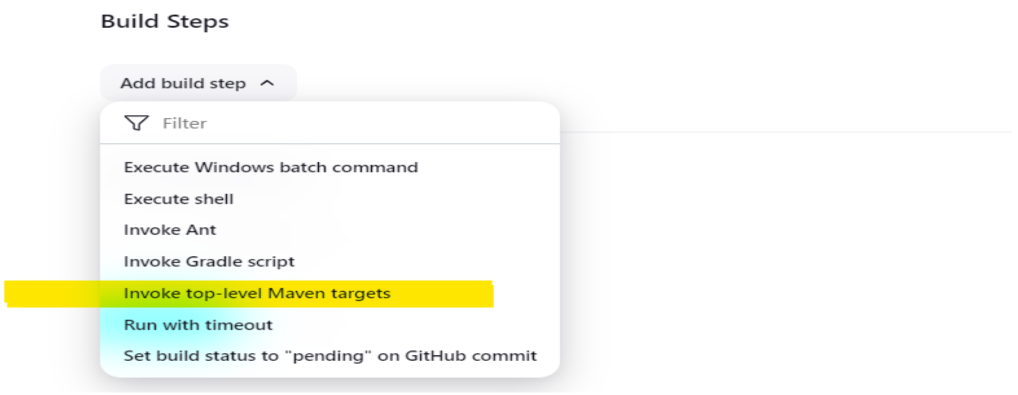
D:\Java-Selenium Course\Automation\SeleniumCaseStudyAutomationProject

Display Name

- [ ] Keep the build logs of dependencies

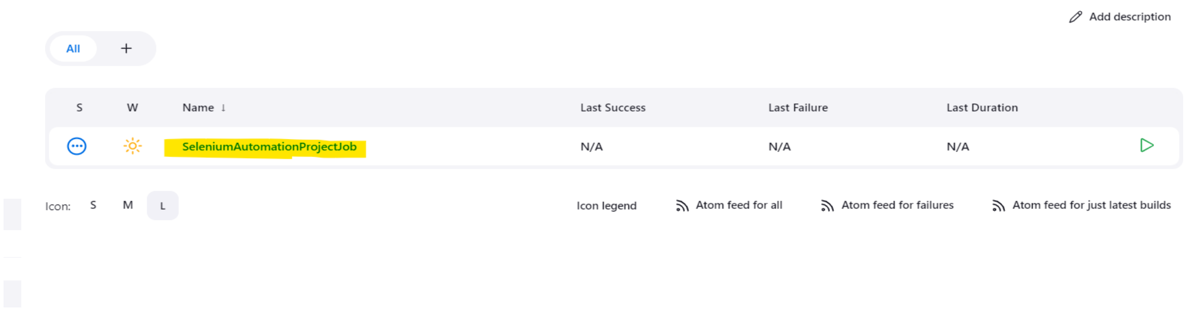
c. Source code management >> None

d. Build Steps >> Add build step >> select 'Invoke top-level Maven targets'


Build Steps
Add build step
Filter
Execute Windows batch command
Execute shell
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Run with timeout
Set build status to "pending" on GitHub commit

- Select Maven Version >> MAVEN_HOME
- Enter **Goal** = test
- Apply and Save

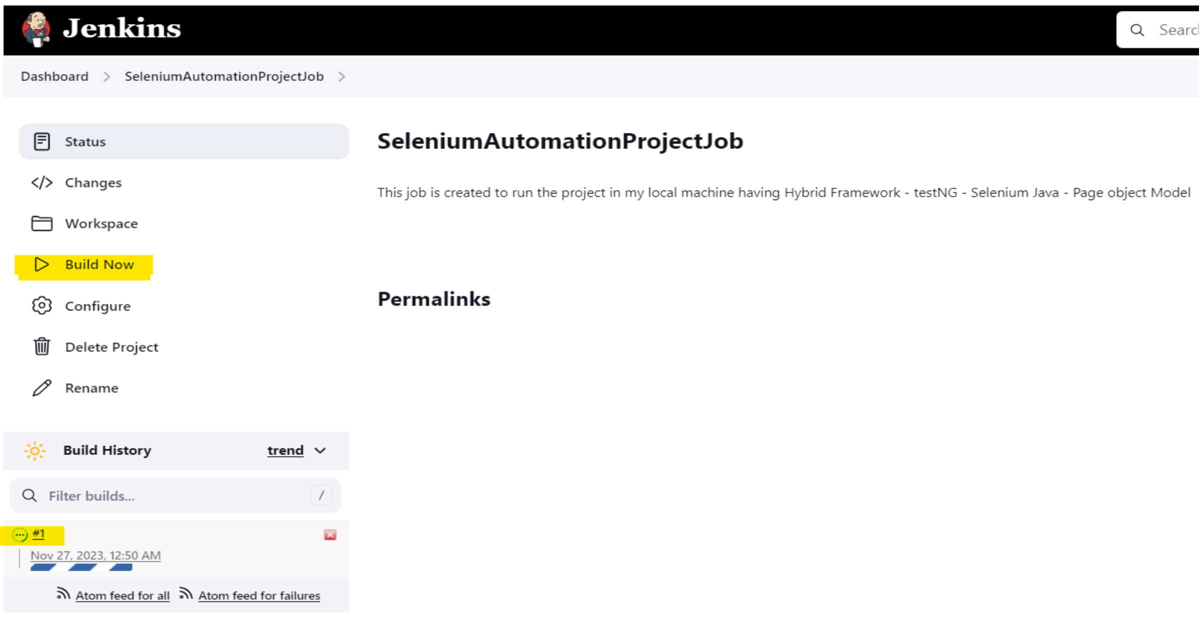
Now go to Dashboard and you will see the job created.



S	W	Name ↓	Last Success	Last Failure	Last Duration
		SeleniumAutomationProjectJob	N/A	N/A	N/A

Now as per the configuration, when we click on **'Build Now'** this job will automatically invoke the maven. Maven will invoke pom.xml. Pom.xml will invoke testing.xml And lastly testing.xml will invoke all the test cases mentioned in testing.xml.

The build will start running – and the test cases will start executing as shown below.



Dashboard > SeleniumAutomationProjectJob >

SeleniumAutomationProjectJob

This job is created to run the project in my local machine having Hybrid Framework - testNG - Selenium Java - Page object Model

Permalinks

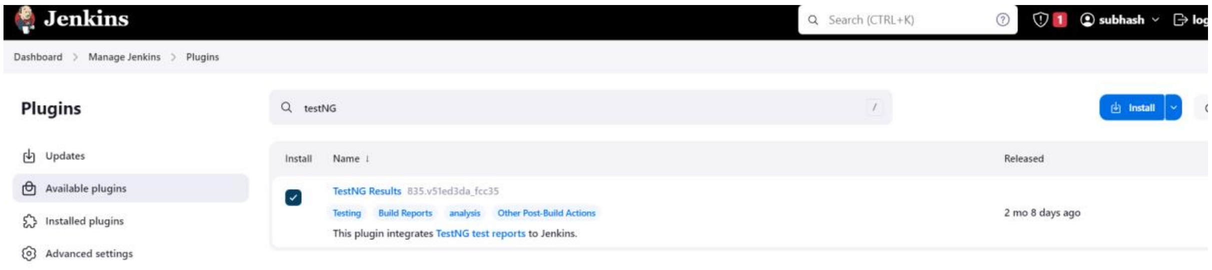
Build History trend

#1 Nov 27, 2023, 12:50 AM

4. TestNG results report in Jenkins

a. Install TestNG

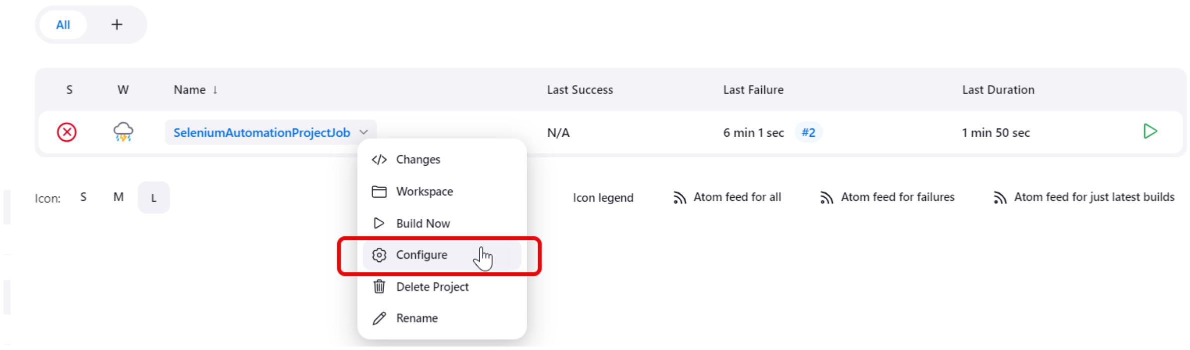
Go to Jenkins Dashboard >> Manage Jenkins >> Plugins >> TestNG >> and click on Install



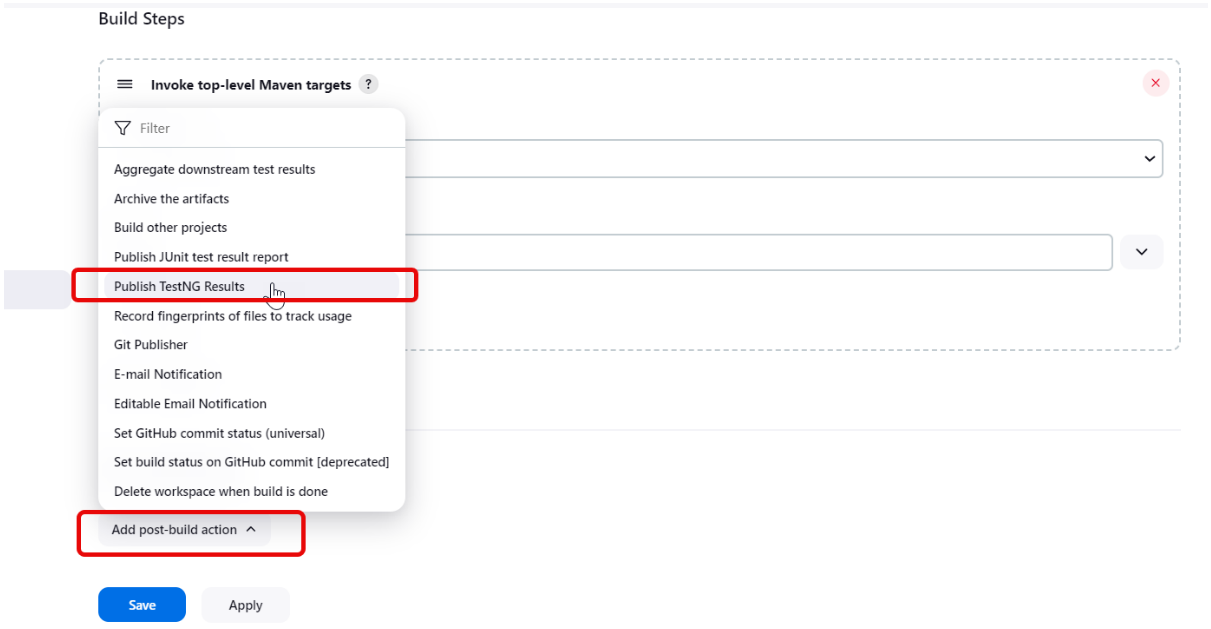
Dashboard > Manage Jenkins > Plugins

Install	Name ↓	Released
✓	TestNG Results 835.v51ed3da_fcc35 Testing Build Reports analysis Other Post-Build Actions This plugin integrates TestNG test reports to Jenkins.	2 mo 8 days ago

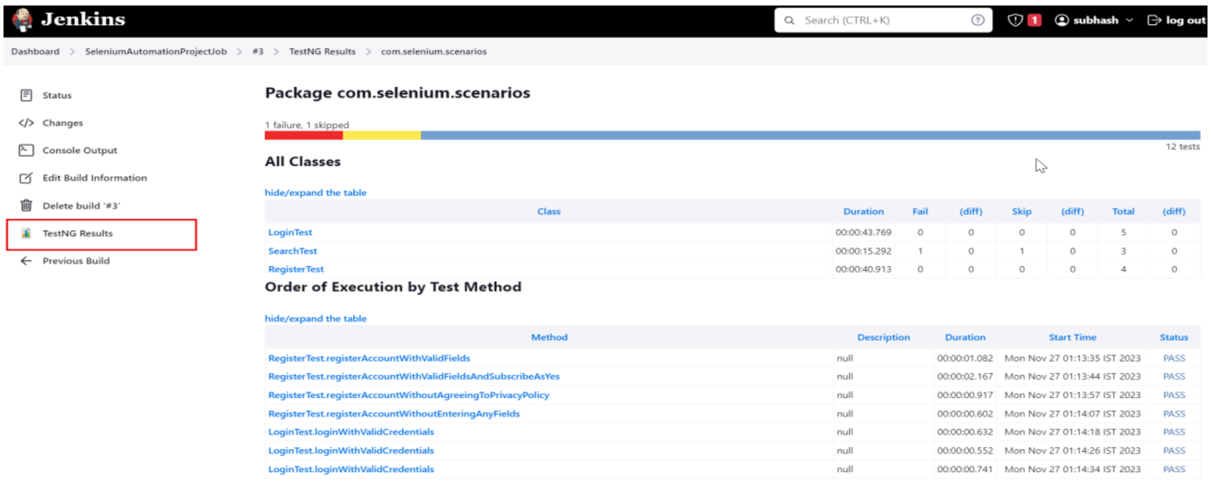
b. Configure TestNG



>> click on 'Add Post-Build Actions' >> select 'Publish TestNG Results' option from dropdown. >> click on 'Apply & Save'


Build Steps
Invoke top-level Maven targets
Filter
Aggregate downstream test results
Archive the artifacts
Build other projects
Publish JUnit test result report
Publish TestNG Results
Record fingerprints of files to track usage
Git Publisher
E-mail Notification
Editable Email Notification
Set GitHub commit status (universal)
Set build status on GitHub commit [deprecated]
Delete workspace when build is done
Add post-build action
Save
Apply

Now go back to dashboard and run the job again >> select the build and we will see the **'TestNG results'** option listed as shown in the screenshot below.



Package com.selenium.scenarios

1 failure, 1 skipped

12 tests

All Classes

hide/expand the table

Class	Duration	Fail	(diff)	Skip	(diff)	Total	(diff)
LoginTest	00:00:43.769	0	0	0	0	5	0
SearchTest	00:00:15.292	1	0	1	0	3	0
RegisterTest	00:00:40.913	0	0	0	0	4	0

Order of Execution by Test Method

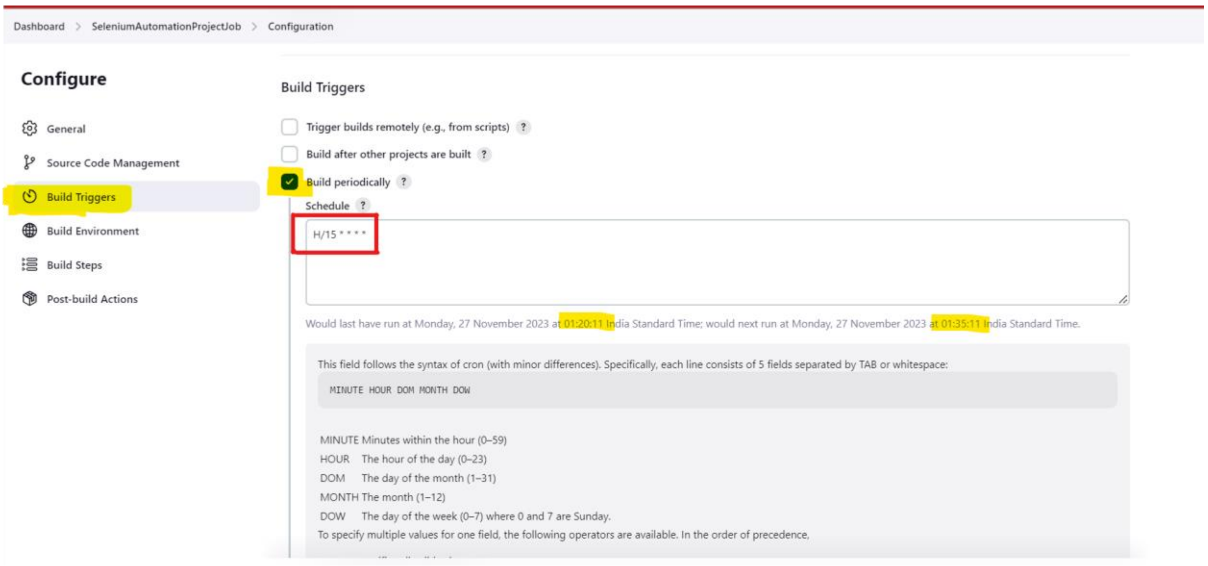
hide/expand the table

Method	Description	Duration	Start Time	Status
RegisterTest.registerAccountWithValidFields	null	00:00:01.082	Mon Nov 27 01:13:35 IST 2023	PASS
RegisterTest.registerAccountWithValidFieldsAndSubscribeAsYes	null	00:00:02.167	Mon Nov 27 01:13:44 IST 2023	PASS
RegisterTest.registerAccountWithoutAgreeingToPrivacyPolicy	null	00:00:00.917	Mon Nov 27 01:13:57 IST 2023	PASS
RegisterTest.registerAccountWithoutEnteringAnyFields	null	00:00:00.602	Mon Nov 27 01:14:07 IST 2023	PASS
LoginTest.loginWithValidCredentials	null	00:00:00.632	Mon Nov 27 01:14:18 IST 2023	PASS
LoginTest.loginWithValidCredentials	null	00:00:00.552	Mon Nov 27 01:14:26 IST 2023	PASS
LoginTest.loginWithValidCredentials	null	00:00:00.741	Mon Nov 27 01:14:34 IST 2023	PASS

5. Build Periodically

>> configure >> Build triggers >> select 'Build Periodically' check box

- This will help us to give Jenkins some instructions to run the automated scripts every 15 minutes. Jenkins will do the work. Basically this will run the scripts periodically.
- Example: H/15 * * * *



Dashboard > SeleniumAutomationProjectJob > Configuration

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

Build Triggers

- [ ] Trigger builds remotely (e.g., from scripts) ?
- [ ] Build after other projects are built ?
- [x] Build periodically ?

Schedule ?

```
H/15 * * * *
```

Would last have run at Monday, 27 November 2023 at 01:20:11 India Standard Time; would next run at Monday, 27 November 2023 at 01:35:11 India Standard Time.

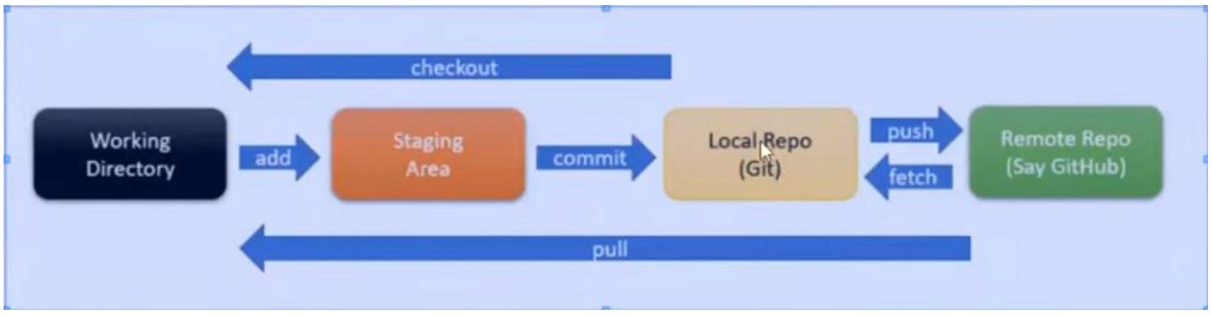
This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:

```
MINUTE HOUR DOM MONTH DOW
```

MINUTE	Minutes within the hour (0–59)
HOUR	The hour of the day (0–23)
DOM	The day of the month (1–31)
MONTH	The month (1–12)
DOW	The day of the week (0–7) where 0 and 7 are Sunday.

To specify multiple values for one field, the following operators are available. In the order of precedence,

Git & GitHub



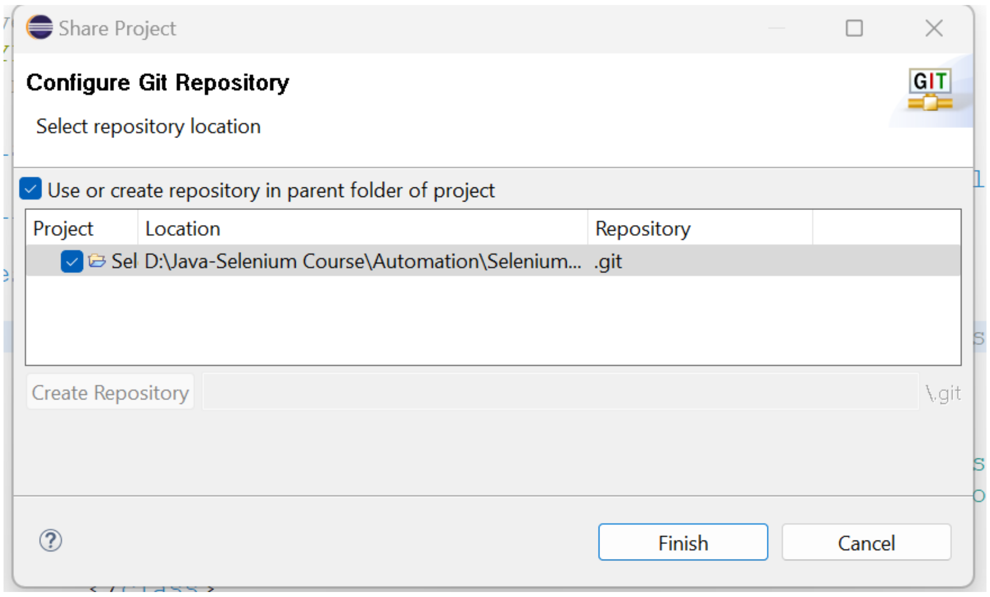
Create Project in eclipse IDE

Create local Git repository in your local machine

- Seach in eclipse for git repository

Now from working directory we need to move the code to Staging Area using add command

- Right click on Project >> Team >> Share Project >> select 'Use or create repository in parent folder of project' check box >> select the project >> click on 'Create repository' button and finish. you will see below screen:



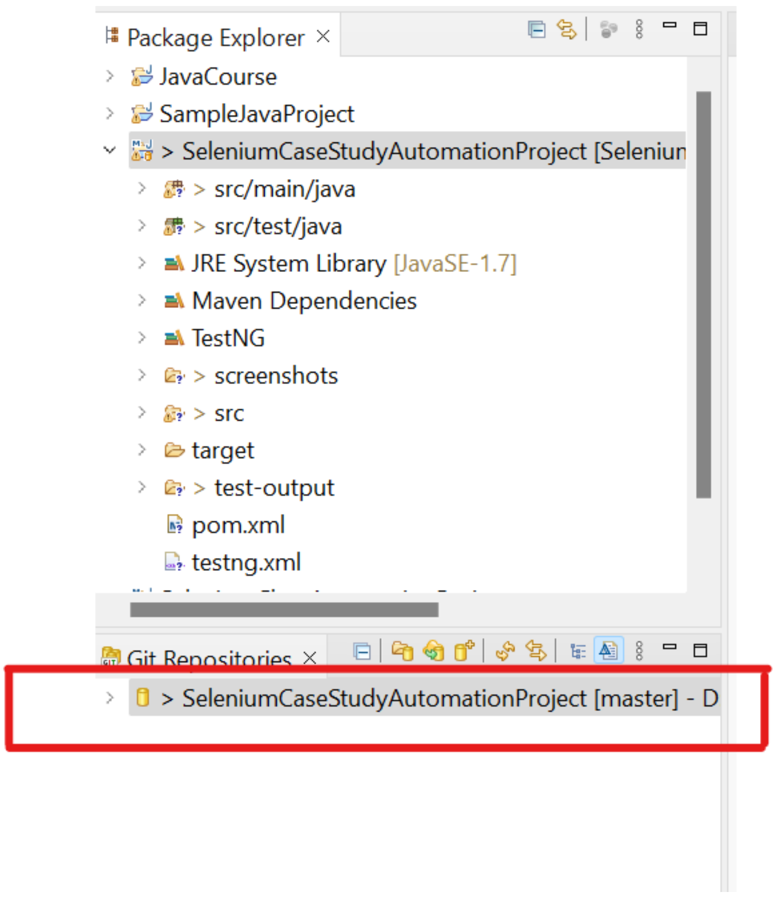
Configure Git Repository

Select repository location

☑ Use or create repository in parent folder of project

Project	Location	Repository
☑ Sel	D:\Java-Selenium Course\Automation\Selenium...	.git

On 'Finish', the project will appear in local git repo, as shown in screenshot below.

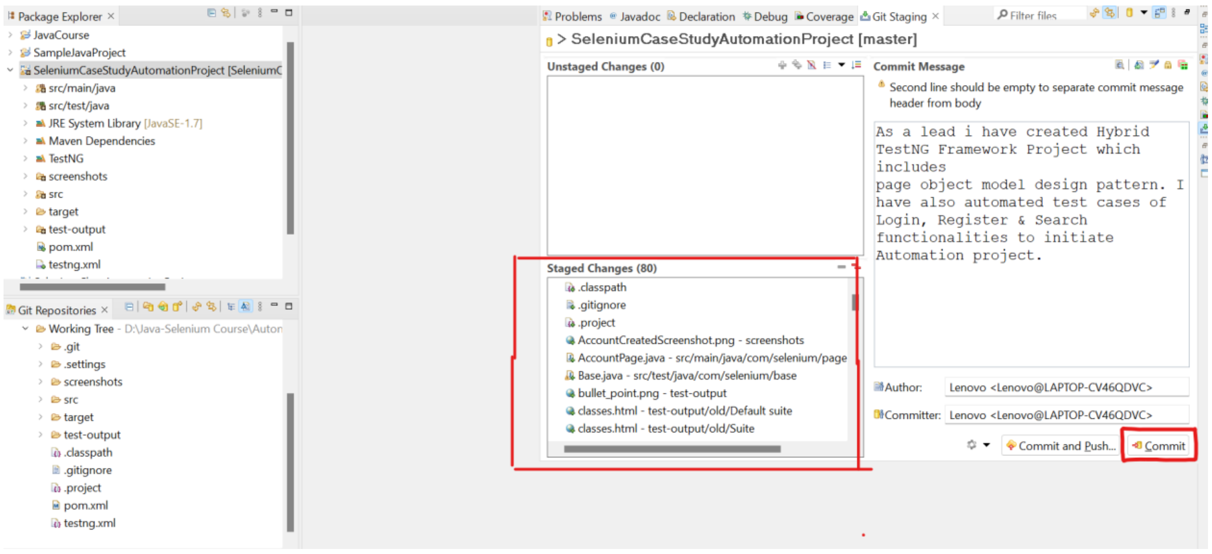


Now we need to move the project code from the working directory to Staging Area:

- Right click on Project >> Team >> '+Add to Index'
- In Staged area the changes are there which needs to be moved in local repo

Now Commit the Project code to the Local Git Repo

- Right click on Project >> Team >> Commit



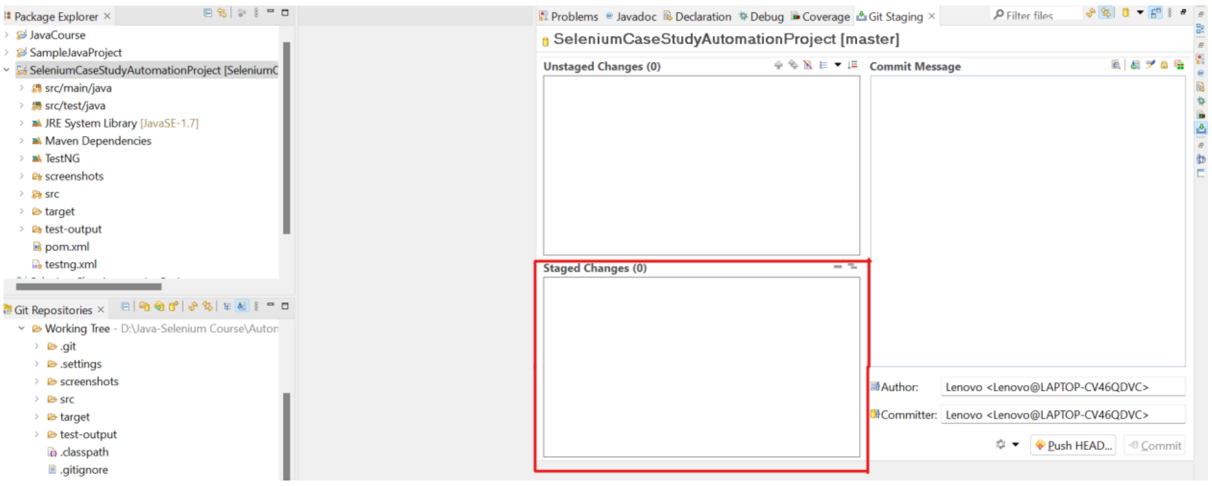
Second line should be empty to separate commit message header from body

```
As a lead i have created Hybrid
TestNG Framework Project which
includes
page object model design pattern. I
have also automated test cases of
Login, Register & Search
functionalities to initiate
Automation project.
```

Author: Lenovo <Lenovo@LAPTOP-CV46QDVC>

Committer: Lenovo <Lenovo@LAPTOP-CV46QDVC>

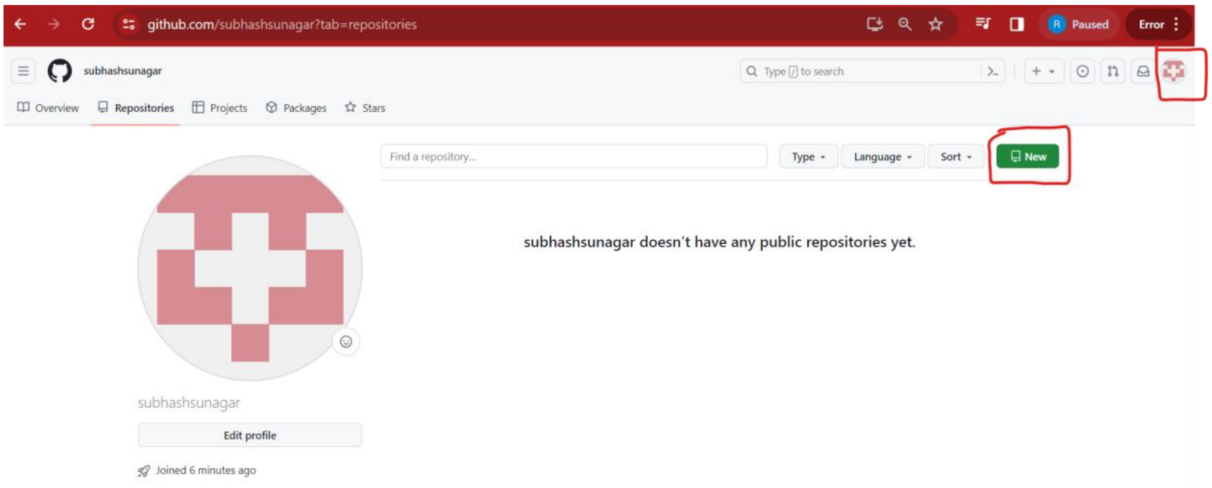
After Successful Commit, the changes from staging area are gone into the local Git repo as shown below:



Now from local Git repo, we need to push the code to Remote repo (GitHub repo)

Go to https://github.com/ and create an Account and click on ‘Your Repositories’ option available from right top corner.

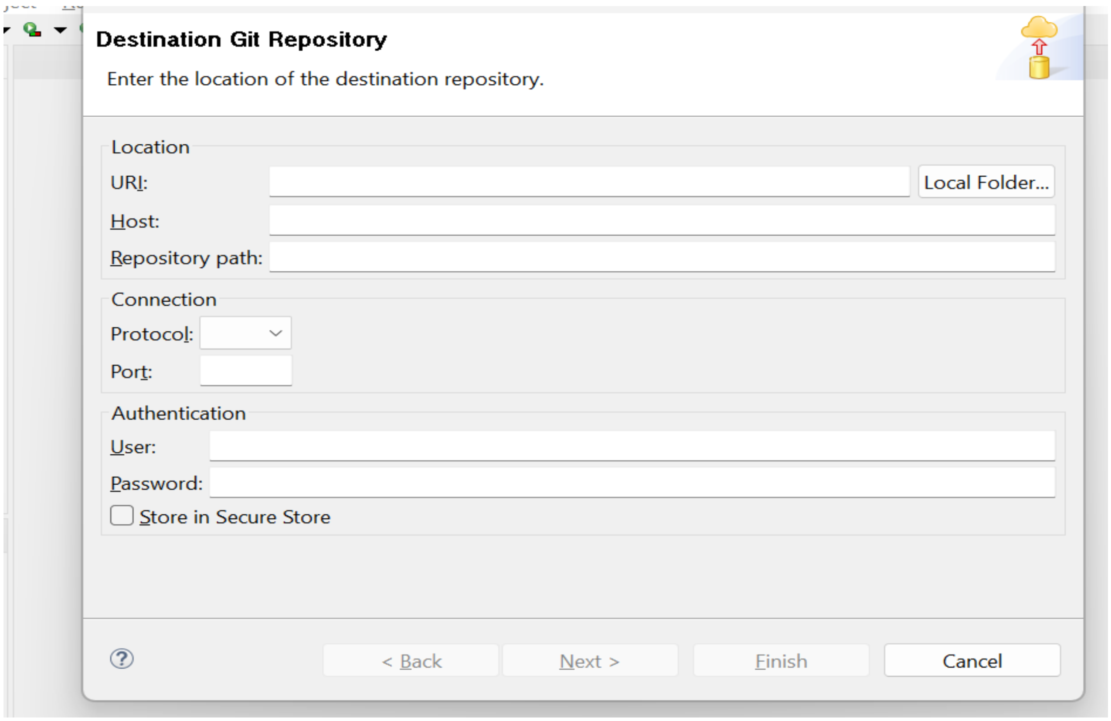
Click on ‘New’ and create a new Repository



Provide ‘Repository Name’ - **AutomationTestingHybridFrameworkRepo** and click on Create Repository.

This is the URL of the repository:
https://github.com/subhashsunagar/AutomationTestingHybridFrameworkRepo.git

Now go to Eclipse >> Right click on Project >> Team >> Remote >> Push


Destination Git Repository

Enter the location of the destination repository.

Location

URI: | Local Folder...

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

User:

Password:

☐ Store in Secure Store

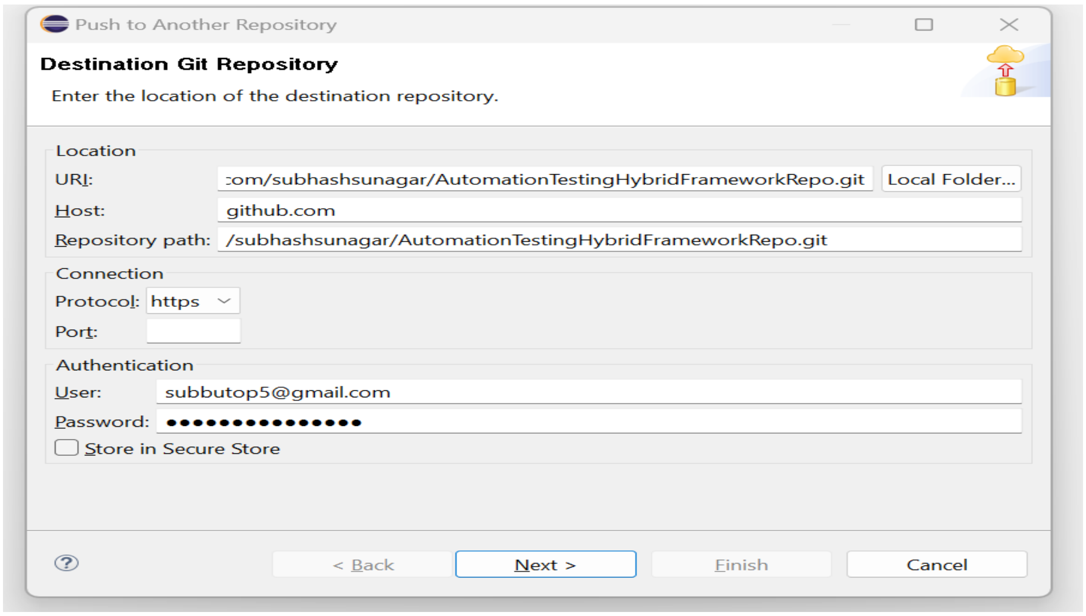
< Back | Next > | Finish | Cancel

Enter the copied Github repo url.

For Authentication, Enter Username and Generate Token code in place of Password

- Generate Token from below url:
- https://github.com/settings/tokens
- click on 'Generate New Token'
- **Token:** is valid for 30 days.

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use this token to login.


Push to Another Repository

Destination Git Repository

Enter the location of the destination repository.

Location

URI: :om/subhashsunagar/AutomationTestingHybridFrameworkRepo.git | Local Folder...

Host: github.com

Repository path: /subhashsunagar/AutomationTestingHybridFrameworkRepo.git

Connection

Protocol: https

Port:

Authentication

User: subbutop5@gmail.com

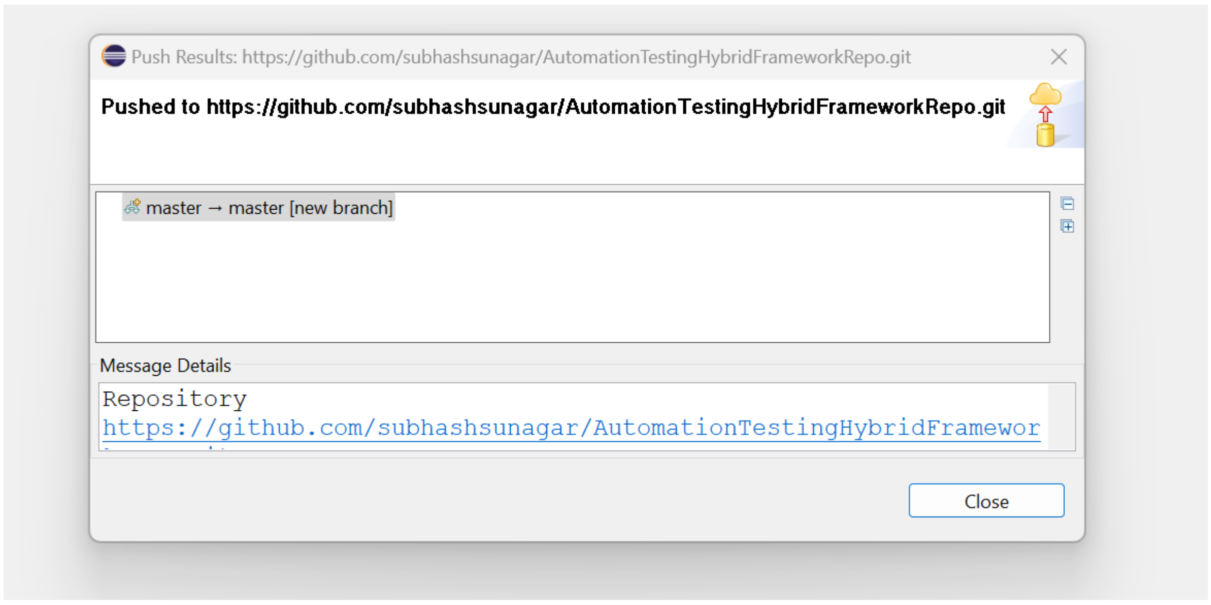
Password: ●●●●●●●●●●●●●●●●

☐ Store in Secure Store

< Back | Next > | Finish | Cancel

>> Select Source Reference as ‘**Master**’ >> click on ‘**+ Add Specs**’ >> Force update >> Finish

Below pop will appear – code has been pushed to your GitHub repository.



Push Results: https://github.com/subhashsunagar/AutomationTestingHybridFrameworkRepo.git

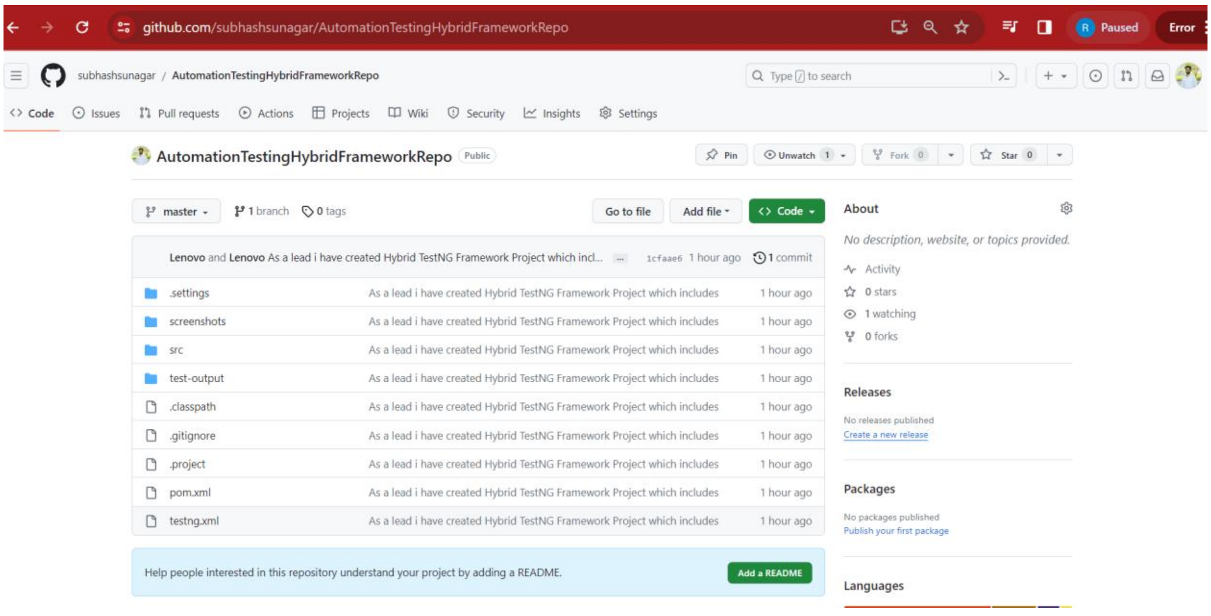
Pushed to https://github.com/subhashsunagar/AutomationTestingHybridFrameworkRepo.git

master → master [new branch]

Message Details

```
Repository
https://github.com/subhashsunagar/AutomationTestingHybridFramewor
```

Refresh the GitHub url and you will see the Project code pushed in your remote Github repo.



github.com/subhashsunagar/AutomationTestingHybridFrameworkRepo

subhashsunagar / AutomationTestingHybridFrameworkRepo

AutomationTestingHybridFrameworkRepo Public

1 branch 0 tags

Lenovo and Lenovo As a lead i have created Hybrid TestNG Framework Project which incl... 1cfaae6 1 hour ago 1 commit

.settings	As a lead i have created Hybrid TestNG Framework Project which includes	1 hour ago
screenshots	As a lead i have created Hybrid TestNG Framework Project which includes	1 hour ago
src	As a lead i have created Hybrid TestNG Framework Project which includes	1 hour ago
test-output	As a lead i have created Hybrid TestNG Framework Project which includes	1 hour ago
.classpath	As a lead i have created Hybrid TestNG Framework Project which includes	1 hour ago
.gitignore	As a lead i have created Hybrid TestNG Framework Project which includes	1 hour ago
.project	As a lead i have created Hybrid TestNG Framework Project which includes	1 hour ago
pom.xml	As a lead i have created Hybrid TestNG Framework Project which includes	1 hour ago
testng.xml	As a lead i have created Hybrid TestNG Framework Project which includes	1 hour ago

Help people interested in this repository understand your project by adding a README. Add a README

About

No description, website, or topics provided.

Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

Jenkins – Git Integration - Run the Scrips from GitHub repository using Jenkins.

- Goto Jenkins >> Manage Jenkins >> Plugin >> install Maven.
- Jenkins Dashboard >> New Item >> **'SeleniumAutomationGitHubRepoJob'** >> Create Jenkins job with Maven Project Option.
- Install Git software in your machine.
    - Go to Jenkins Dashboard >> Manage Jenkins >> Tools >> Git Installations >>

      Name = Git
      Path to Git Executable = C:\Program Files\Git\bin\git.exe

- Jenkins Dashboard >> Created Job >> Configuration >> Source Code Management >> Select **Git >>**
    - Provide repository url: https://github.com/subhashsunagar/AutomationTestingHybridFrameworkRepo
- Pre Steps >> Invoke top-level Maven targets >> Provide detail >>
    - Maven Version = **MAVEN_HOME** &
    - Goals = **test**
- Post Build Actions >> Publish TestNG Results
- Apply and Save