Selenium Tutorial for Beginners using Python

Introduction to Selenium

Selenium is a free (open-source) automated testing suite for web applications that supports cross-browser and cross-operating-system interoperability. It is quite similar to HP's QuickTest Pro (QTP, now UFT), only that Selenium focuses on automating web-based applications. Testing done using the Selenium tool is usually referred to as Selenium Testing.

Selenium is useful for testing web applications only. Neither desktop (software) testing nor the testing of mobile applications is possible with Selenium.

Watch this Selenium Tutorial for Beginners

A web application is an application program stored on a remote server that is allowed to get accessed through a web browser over the Internet. Many websites contain web applications. Any website component that performs functions for users qualifies as a web application. Few examples of web applications are:

- Google Search Engine
- AliExpress

Coming back, the testing of web applications done through the Selenium tool is referred to as Selenium Testing.

Selenium Types

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

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

Now that you know about its types, let's talk about each of them briefly.

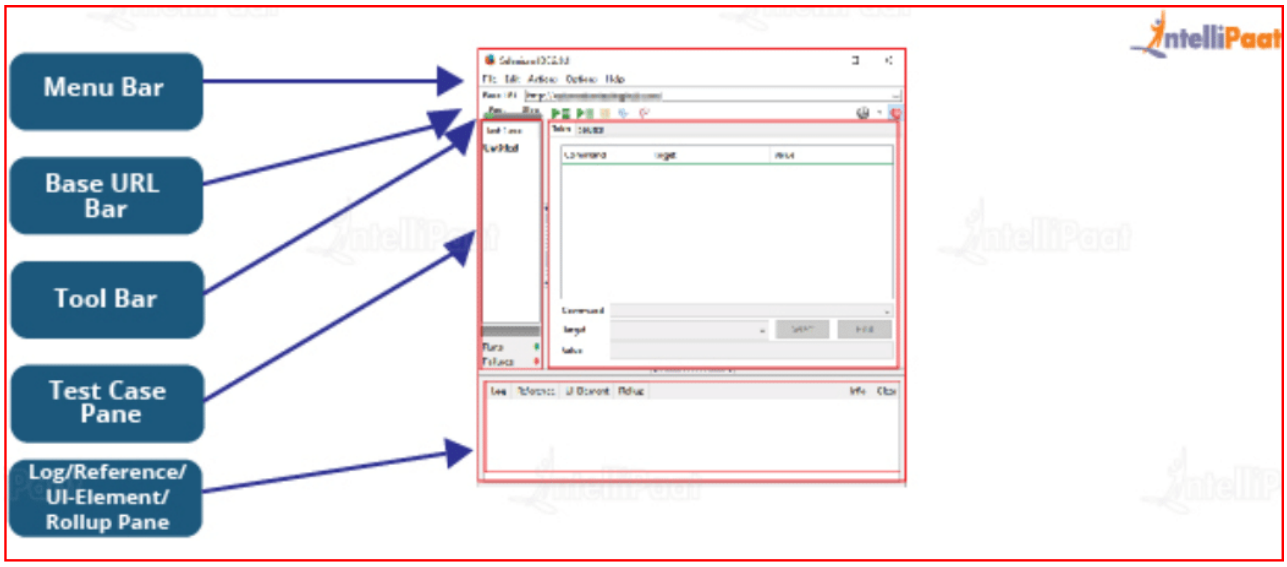
Selenium IDE

Selenium IDE (Integrated Development Environment) is a tool that helps you develop your Selenium test cases. It's an easy-to-use Chrome and Firefox extension and is generally the most reliable method to develop test cases. It records users' actions in the browser for you, using the existing Selenium commands, with parameters defined by the context of the web element. This is not only a time-saver but also an excellent way of learning Selenium script syntax. Previously known as Selenium Recorder, Selenium IDE was initially created by Shinya Kasatani, of Japan and contributed to the Selenium project in 2006.

It was introduced as a Firefox plugin for faster creation of test cases. As it was a Firefox extension, it could automate the browser through a record-and-play feature providing autocompletion support and the ability to move commands around quickly.

Scripts are recorded in a special test scripting language called Selenese for Selenium. Selenese comes up with commands for carrying out actions in a web browser and restoring data from the resulting pages.

The advantage of Selenium IDE is that the tests recorded via the plugin can be exported in different programming languages like Java, Ruby, Python, etc.



Selenium RC (Remote Control)

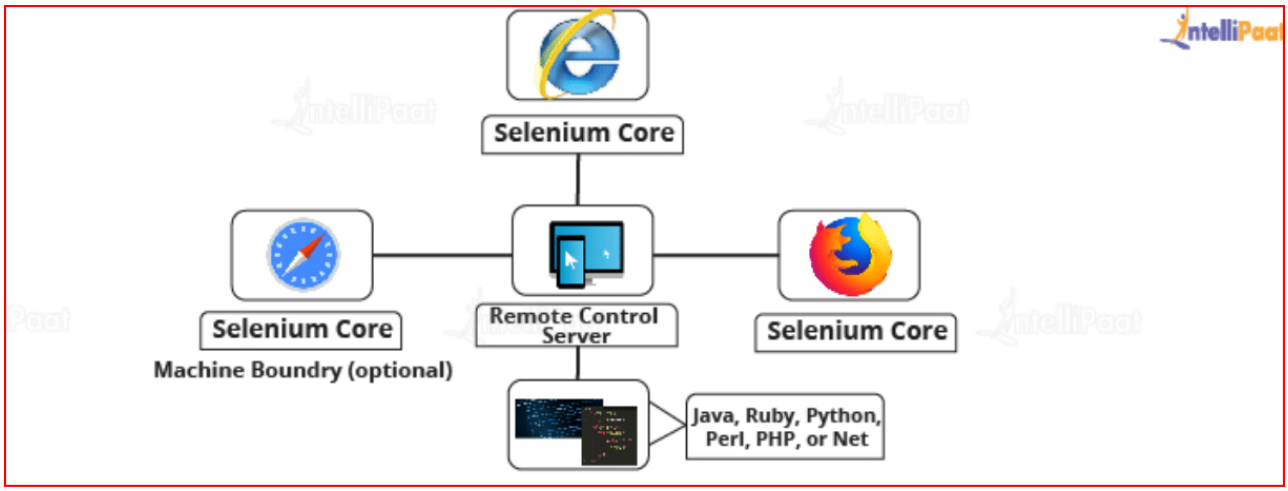
Let me first tell you that Selenium Core was the first version. But with that version, testers had to install both Selenium (a JavaScript program) and the webserver containing the web application being tested on their local systems so that they would belong to the same domain.

Then, another ThoughtWorks' engineer, Paul Hammant decided to create a server that will act as an HTTP proxy to trick the browser into believing that Selenium Core and the web application being tested belong to the same domain, thus making RC a two-component tool.

- Selenium RC Server
- Selenium RC Client (*Library containing the programming language code)*

RC can support the following programming languages:

- Java
- C#
- PHP
- Python
- Perl
- Ruby

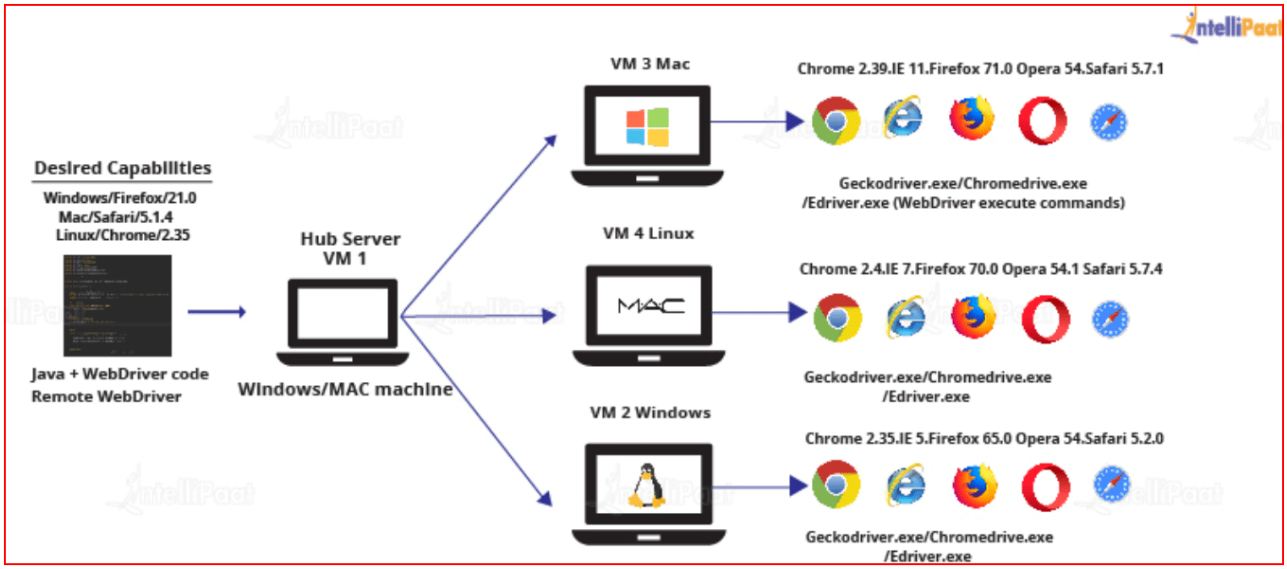


Selenium Grid

Selenium Grid is a testing tool that lets you run your tests on various machines against different browsers. It is part of the Selenium suite that specializes in running multiple tests across different browsers, operating systems, and machines. You can connect to it with Selenium Remote Control by stating the browser version, browser, and operating system as per your choice. You will be able to specify these values through Selenium Remote Control capabilities. With Selenium Grid, one server makes a move as a hub. Tests communicate to the hub to get access to browser instances. The hub has a list of servers that provide access to browser instances (WebDriver nodes) and lets tests use these instances.

Selenium Grid allows parallel testing and also allows managing different browser versions and browser configurations centrally (instead of in each individual test).
There are multiple online platforms that provide an online Selenium Grid that you can access to run your Selenium automation scripts. For example, you can use LambdaTest.

Selenium Grid has more than 2,000 browser environments over which you can run your tests and truly automate cross-browser testing.



Selenium WebDriver

Founded by Simon Stewart in 2006, ThoughtWorks consultant in Australia. Selenium WebDriver was the first cross-platform testing framework that would control the browser at the OS level. Selenium WebDriver is a successor to Selenium RC. Selenium WebDriver accepts commands (sent in Selenium or via a Client API) and sends them to a browser.

This is implemented through a browser-specific driver, which sends commands to a browser and retrieves the results. Each driver launches and accesses a browser application. Different WebDrivers are:

- Firefox Driver (Gecko Driver)
- Chrome Driver
- Internet Explorer Driver
- MicroEdge
- Opera Driver
- Safari Driver
- HTML Unit Driver

Benefits of Selenium WebDriver

- Selenium WebDriver supports seven programming languages: Java, C#, PHP, Ruby, Perl, Python, and .Net.
- It supports cross-browser interoperability that helps you perform testing on various browsers like Firefox, Chrome, IE, Safari, etc.
- Tests can be performed on different operating systems: Windows, Mac, Linux, Android, and iOS.
- Selenium WebDriver overcomes limitations of Selenium v1 like file upload, download, pop-ups, and dialog barrier

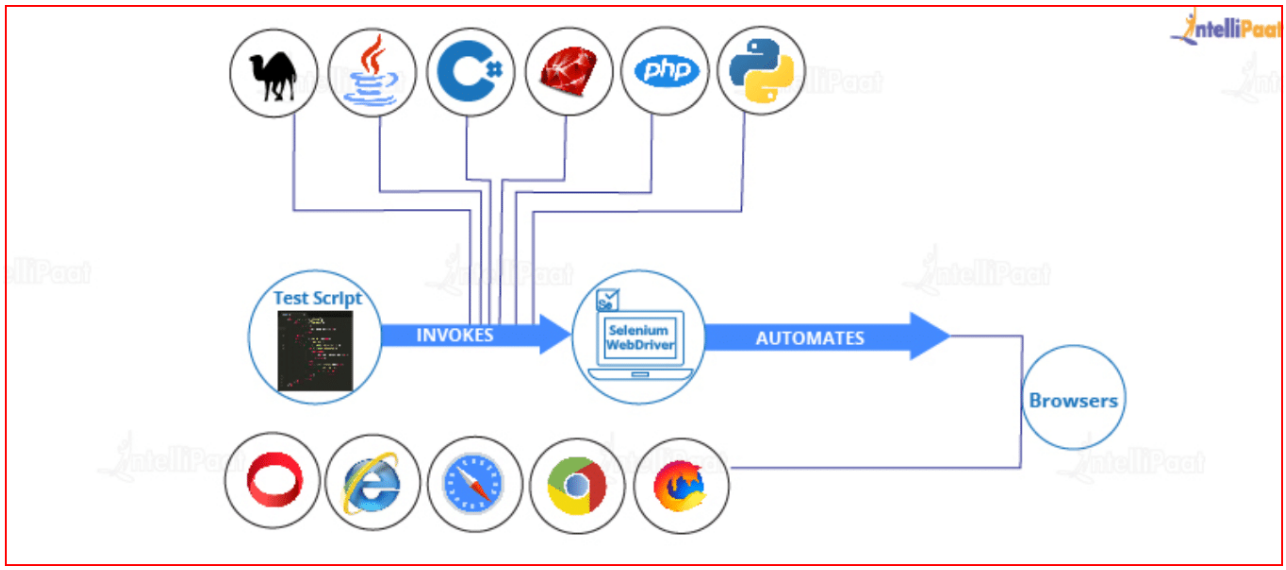
Cons of Selenium WebDriver

- Detailed test reports cannot be generated.
- Testing images is not possible.

No matter what, these shortcomings can be overcome by integrations with other frameworks. That is, for testing images Sikuli can be used, and for generating detailed test reports, TestNG can be used.

Now, you know what Selenium is and have a fair idea about the various tools of the Selenium suite.

Next, I will tell you everything you need to know to get started with testing web apps using Selenium WebDriver. The below image depicts how WebDriver works:



What is Selenium WebDriver?

In this part, let's dig deeper into learning Selenium WebDriver. Let's understand more about what Selenium WebDriver is, what browser elements are, and how to locate browser elements on a web page.

Locating and testing of web elements in the web application is implemented through a browser-specific driver. It controls the browser by directly communicating with it.
In Selenium WebDriver, you have the liberty to write test scripts in different programming languages like Java, Perl, Python, Ruby, C#, PHP, and JavaScript. But, make a note that Mozilla Firefox is Selenium WebDriver's default browser.

WebDriver was introduced as part of Selenium v2.0. Selenium v1.0 consisted of only IDE, RC, and Grid. But the major breakthrough in the Selenium project was when WebDriver was developed and introduced as an addition in Selenium v2. And, with the release of Selenium v3, RC has been deprecated and moved to a legacy package. Although you can still download webdriver and carry out tasks with RC, there wouldn't be any support for it in Selenium v3.

In a nutshell, the advantages WebDriver has over RC are:

- Support for more programming languages, operating systems, and web browsers
- Ability to overcome the limitations of Selenium v1
- Simpler commands when compared to RC, and a better API
- Support for batch testing, cross-browser testing, and data-driven testing

The drawback WebDriver has when compared to RC is that test reports cannot be generated in WebDriver; whereas, RC generates detailed reports.
You must have heard the term 'browser elements' a number of times. The next part of this Selenium tutorial will be about these elements, and you will see how testing happens on these web elements.

What are Browser Elements?

Browser elements are different components present on web pages. The most common elements you will notice while browsing are:

- Text boxes
- CTA buttons
- Images
- Hyperlinks
- Radio buttons/Checkboxes
- Text area/Error messages
- Dropdown box/List box/Combo box
- Web table/HTML table
- Frame

Testing these elements essentially means, you have to check whether they are working fine and responding the way you want them to. For example, if you are testing a text box, what would you test it for?

1. Whether you are able to send text or numbers to the text box
2. If you can retrieve text that has been passed to the text box, etc.

If you are testing an image, you might want to:

- Download the image
- Upload the image
- Click on the image link
- Retrieve the image title, etc.

Similarly, operations can be performed on each of the elements mentioned earlier. But only after the elements are located on the web page, you can perform operations on them and start testing them.

Watch this Selenium Tutorial for Beginner

Locating Browser Elements Present on a Web Page

Each element on a web page does have an attribute (property). Elements can have multiple attributes and most of these attributes will be distinctive for different elements. Consider an example, there is a page having two elements: an image and a text box. Both these elements have a 'Name' attribute and an 'ID' attribute. These attribute values need to be distinctive for each of these elements. In other words, no two elements can have the same attribute value.

In the above example, the image and the text box can have neither the same 'ID' nor the same 'Name' value. However, there are some attributes that can be common for a group of elements on the page, like a group of elements, can have the same value for 'Class Name'.

There are eight attributes that can be used to locate elements on a web page, they are ID, Name, Class Name, Tag Name, Link Text, Partial Link Text, CSS, and XPath. Since the elements are located using these attributes, they are referred to as 'Locators'. The locators are:

- By.id

```
driver.findElement(By.id("xxx"));
```

- By.name

```
driver.findElement(By.name("xxx"));
```

- By.className

```
driver.findElement(By.className(“xxx”));
```

- By.tagName

```
driver.findElement(By.tagName(“xxx”));
```

- By.linkText

```
driver.findElement(By.linkText(“xxx”));
```

- By.partialLinkText

```
driver.findElement(By.partialLinkText(“xxx”));
```

- By.css

```
driver.findElement(By.css(“xxx”));
```

- By.xpath

```
driver.findElement(By.xpath(“xxx”));
```

By looking at the syntax above, you might have realized why locators are called inside methods.

Now, you need to learn all the other methods, browser commands, and functions that can be used to perform operations on the elements.
But before moving on with the hands-on, creating an automated test, let’s first understand what dependencies are and how they help you in creating a Maven project.

You will need certain dependencies and libraries ready with you for the Selenium project which will help you perform automated testing of a web application and such a tool is known as Maven.

What is Maven in Selenium?

Maven is a build automation tool used primarily for Java projects by downloading its dependencies.

Basically, Maven is a software that helps you download dependencies for a software program. When you create a Selenium project, you need to specify all Selenium components which are required to be included inside a POM file for the Selenium project to be ready. Once the dependencies are added in the POM file, you can simply save the project and all these dependencies will automatically be downloaded.

Setting up Selenium with Maven and TestNG on Eclipse

Before diving into this section of the Selenium tutorial, let’s see how Eclipse projects run. For making Eclipse Java projects run, you need a library that gives the ability to produce an HTML report of execution and display the test case that has failed, which is done by TestNG library. When bugs can be accurately located like this, they can be fixed immediately to the relief of developers.

TestNG is a testing framework. It structures, groups, and launches tests. It also generates testing reports. To get a function executed, we need to include @Test annotation before the definition of that function.

When you run this file as TestNG suite, the execution will start and you will get the detailed test reports. You will get the test output in your console tab and the result of the test suite in the next tab.

You decided to use TestNG for several reasons:
1. TestNG annotations are easy-to-create test cases.
2. Test cases can be grouped and prioritized more easily.
3. TestNG supports parameterization.
4. It also supports data-driven testing using data providers.
5. It generates HTML reports.
6. Parallel test execution is possible.
7. TestNG readily supports integration with other tools and plugins like Eclipse IDE and build tools like ANT, Maven, etc.

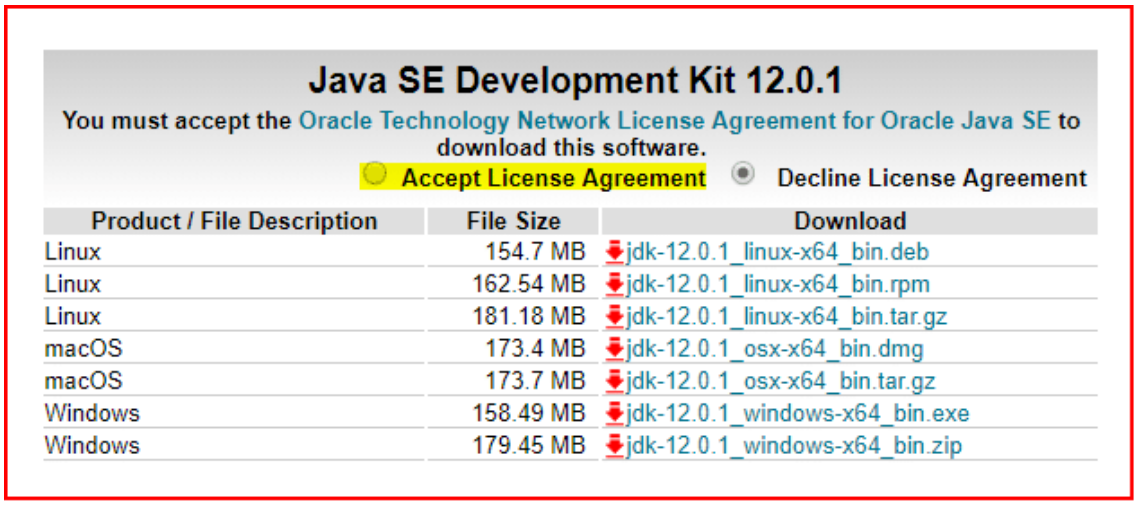
Now, this tutorial will move on with the hands-on part. This section is divided into three parts:

- Java JDK Installation
- Eclipse Installation
- Performing Selenium Test Case

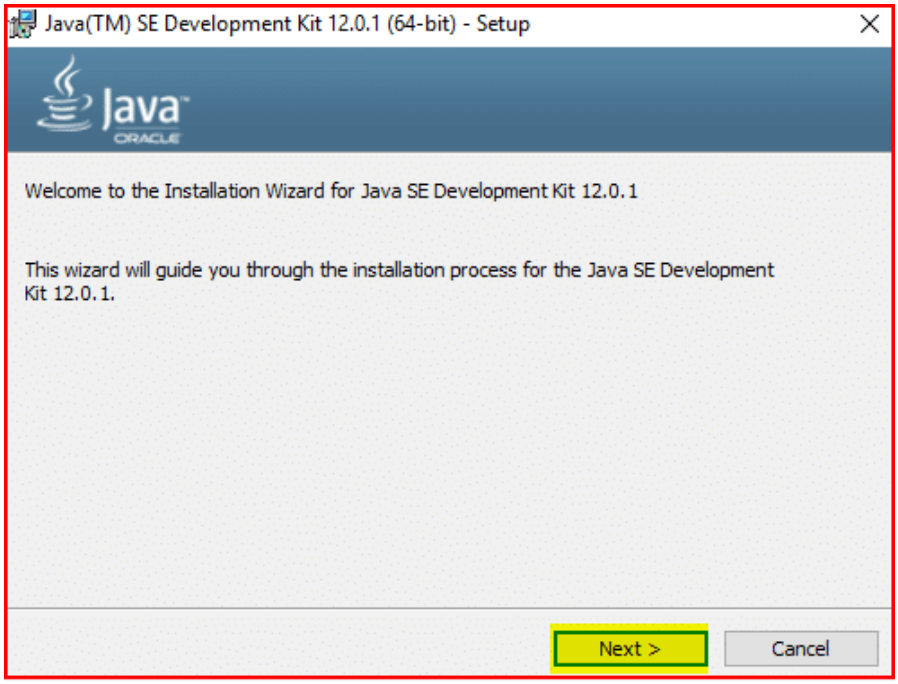
Installing Java JDK

Step 1: Download Java JDK from the link provided below and then click on the Oracle JDK **Download** button
https://www.oracle.com/technetwork/java/javase/downloads/index.html

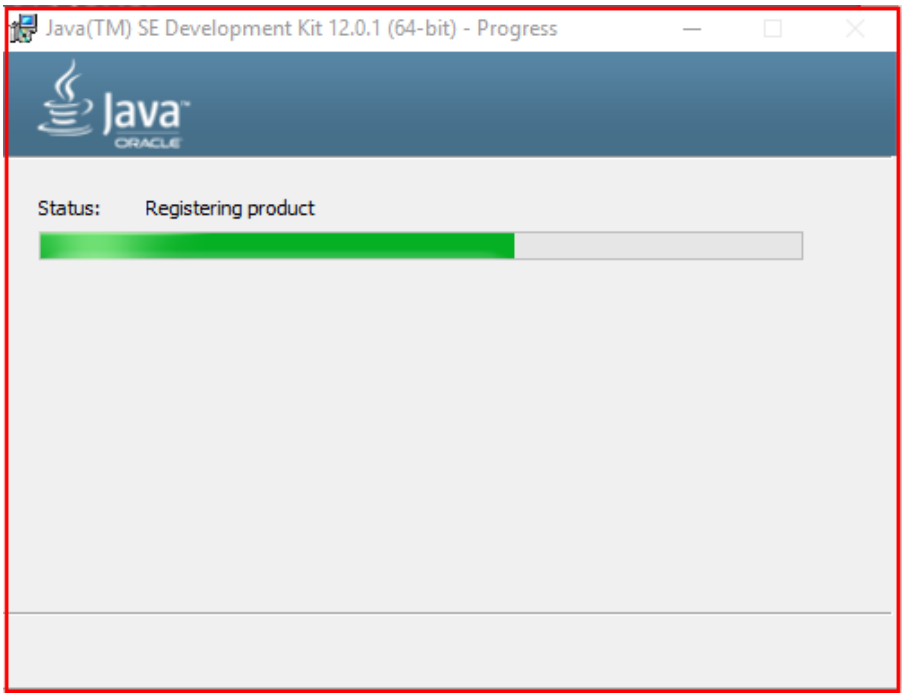
Click on the radio button, **Accept License Agreement**, and download the ***.exe*** file related to your OS type



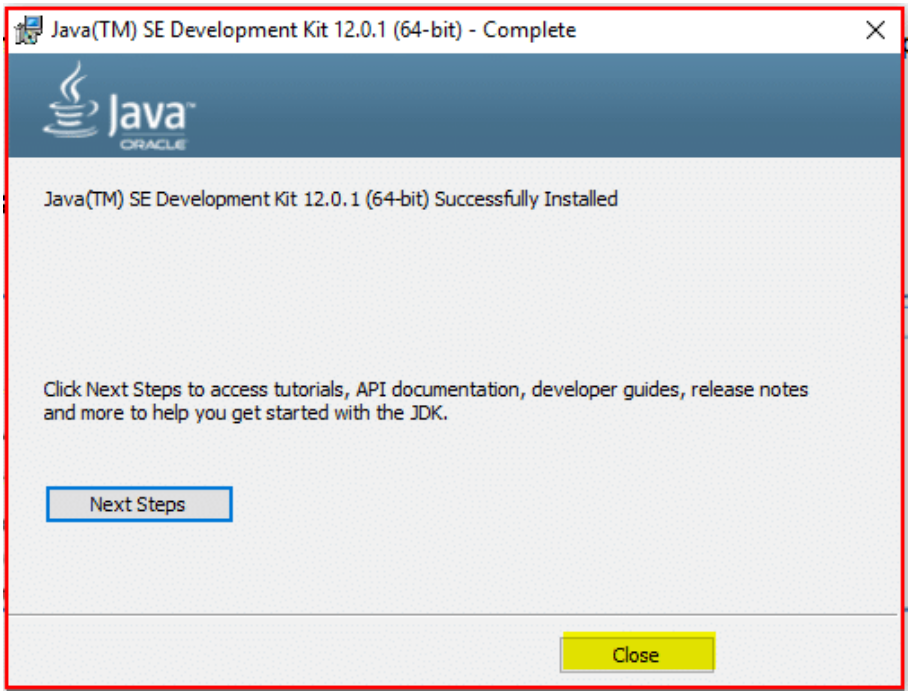
Step 2: Open the downloaded execution file, and click on **Next**



Step 3: Select **Developers Tools** and click on **Next**, and you will be directed to the below screen. It will take a few moments to set up JDK in your system

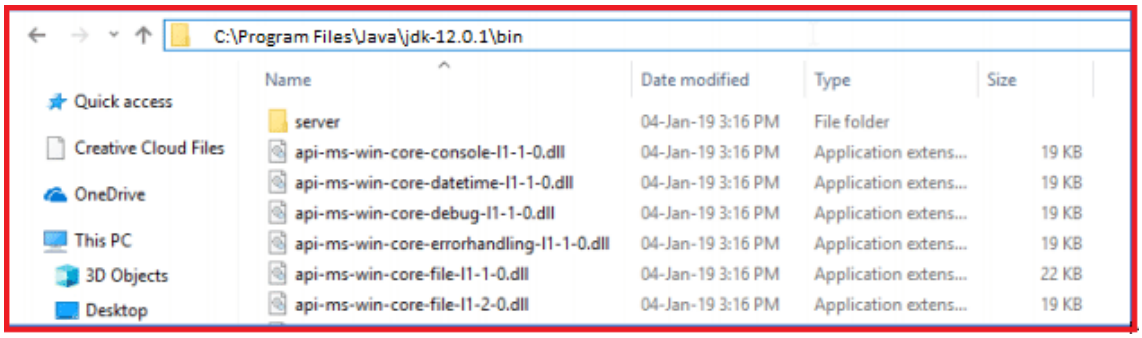


Step 4: Click on **Close** to complete the set up

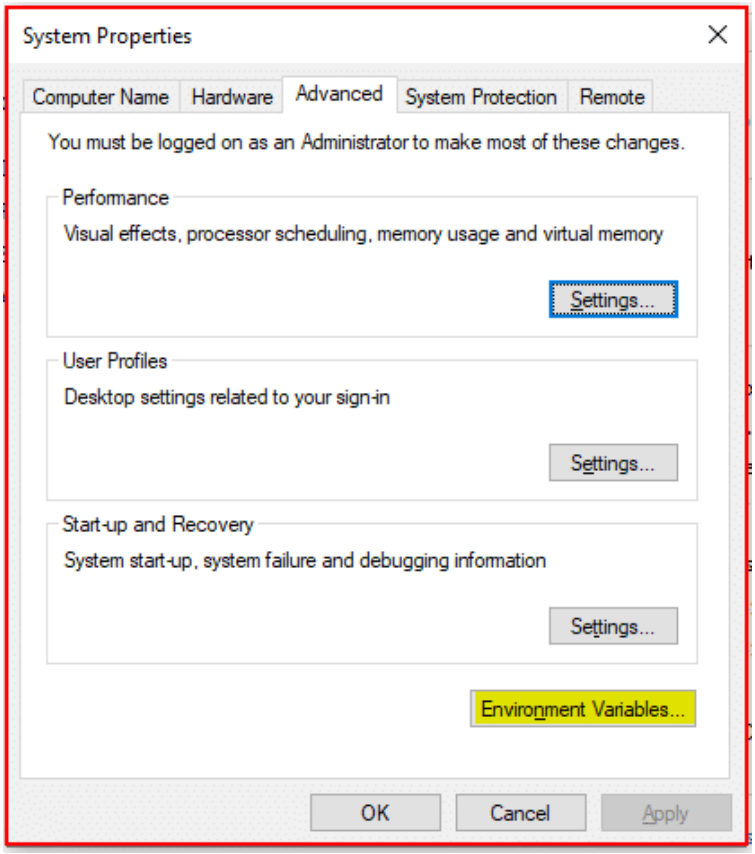


Before moving ahead, you need to add the environment variables to the path as follows.

Step 5: Go to local **C drive** > **Program Files** > **Java** > **jdk-12.0.1** > **bin**

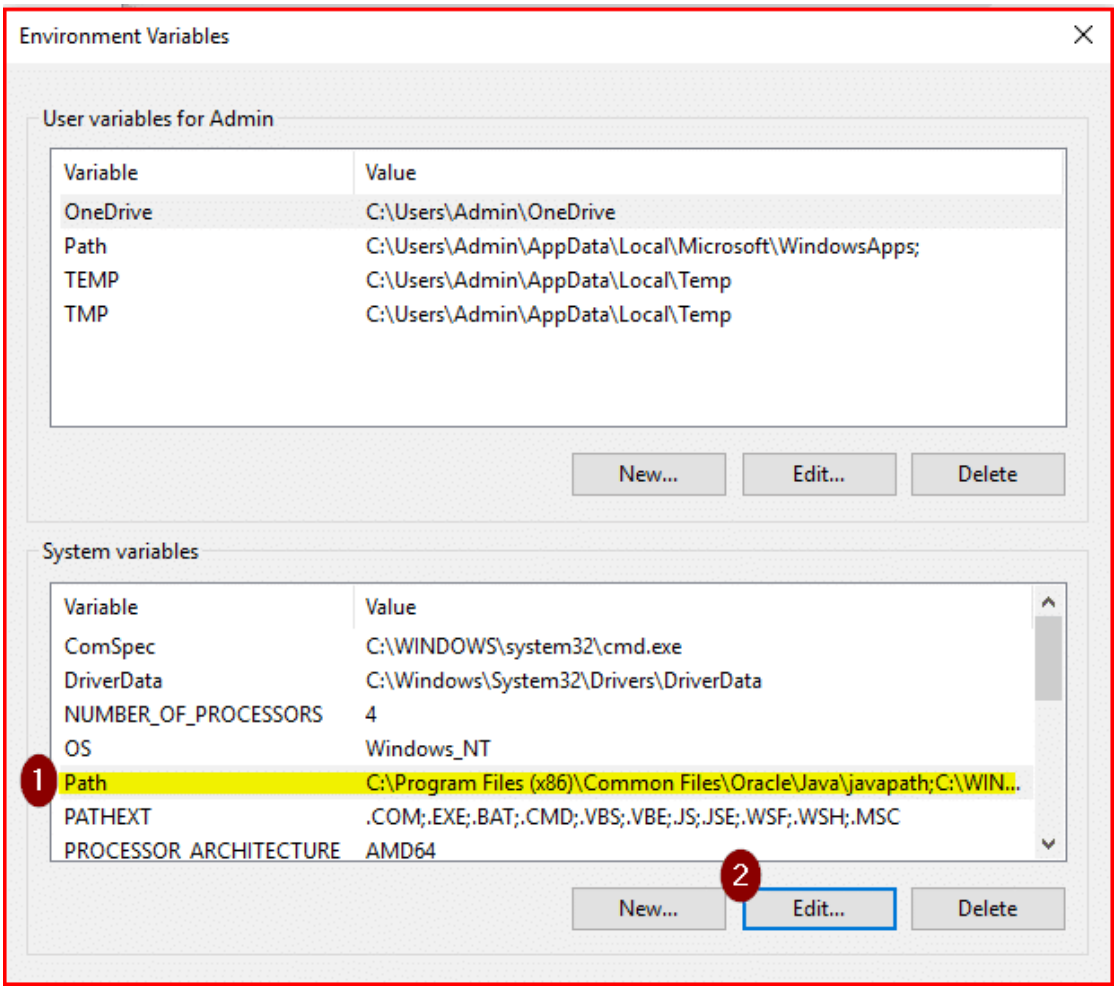


Step 6: Copy the path as shown above
Step 7: Go to **Control Panel** > **System and Security** > **System** > **Advanced system settings** > **System Properties**



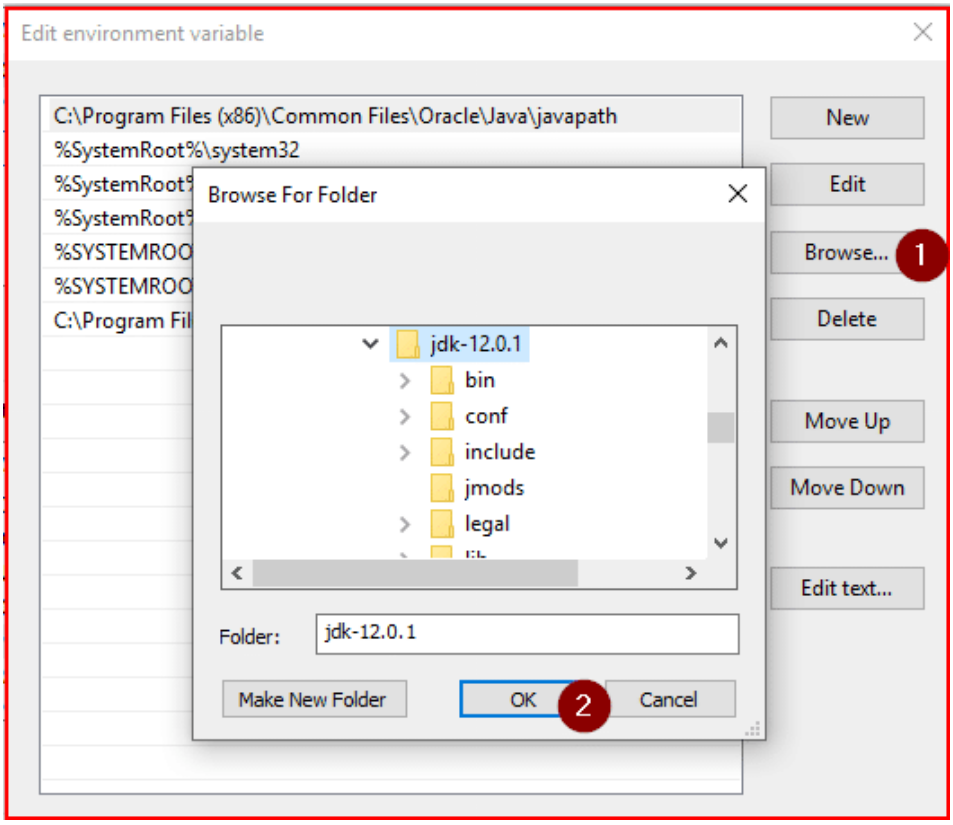
Step 8: Click on **Environment Variables** as shown above and the below window pops up in which you will perform the following steps:

1. Click on **Path** in the **System variables** section, under **Environment Variables**
2. Click on **Edit**



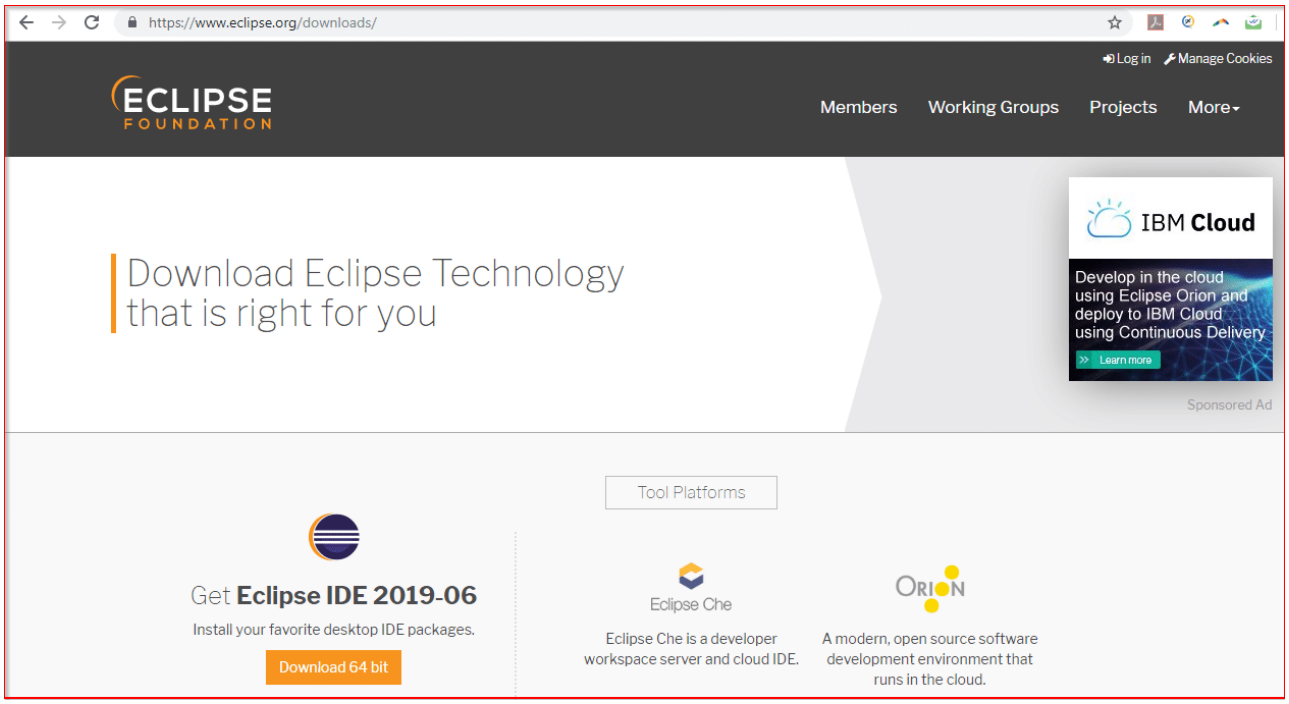
Step 9:

1. Click on **Browse** and find your file path as local C drive > Program Files > Java > jdk-12.0.1 Keep clicking 'OK'
2. Now, you have Java JDK installed in your system

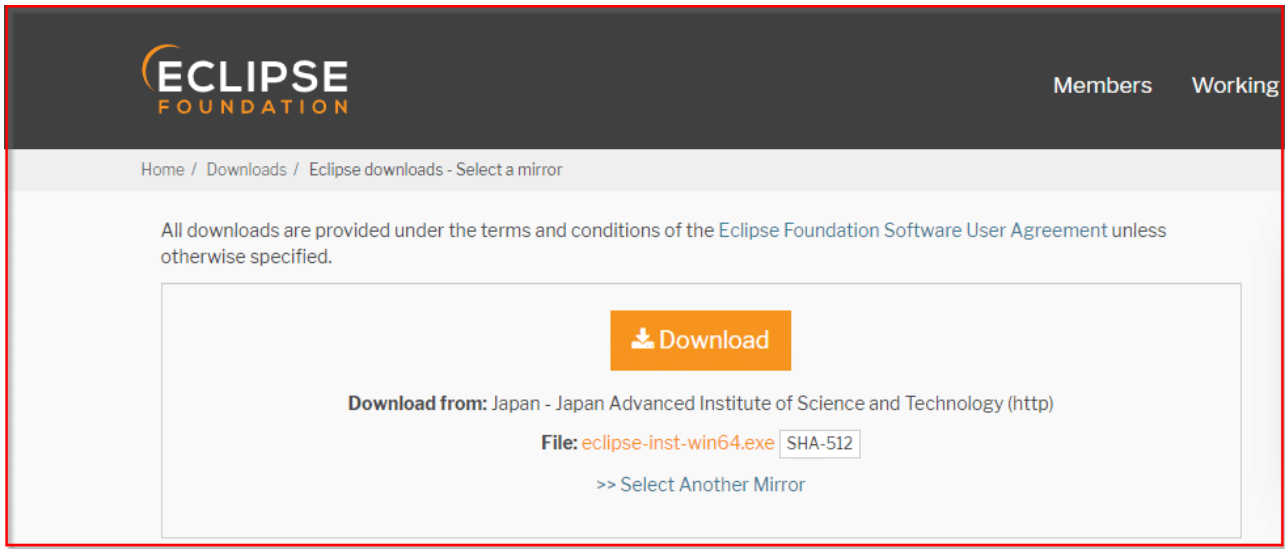


Installing Eclipse

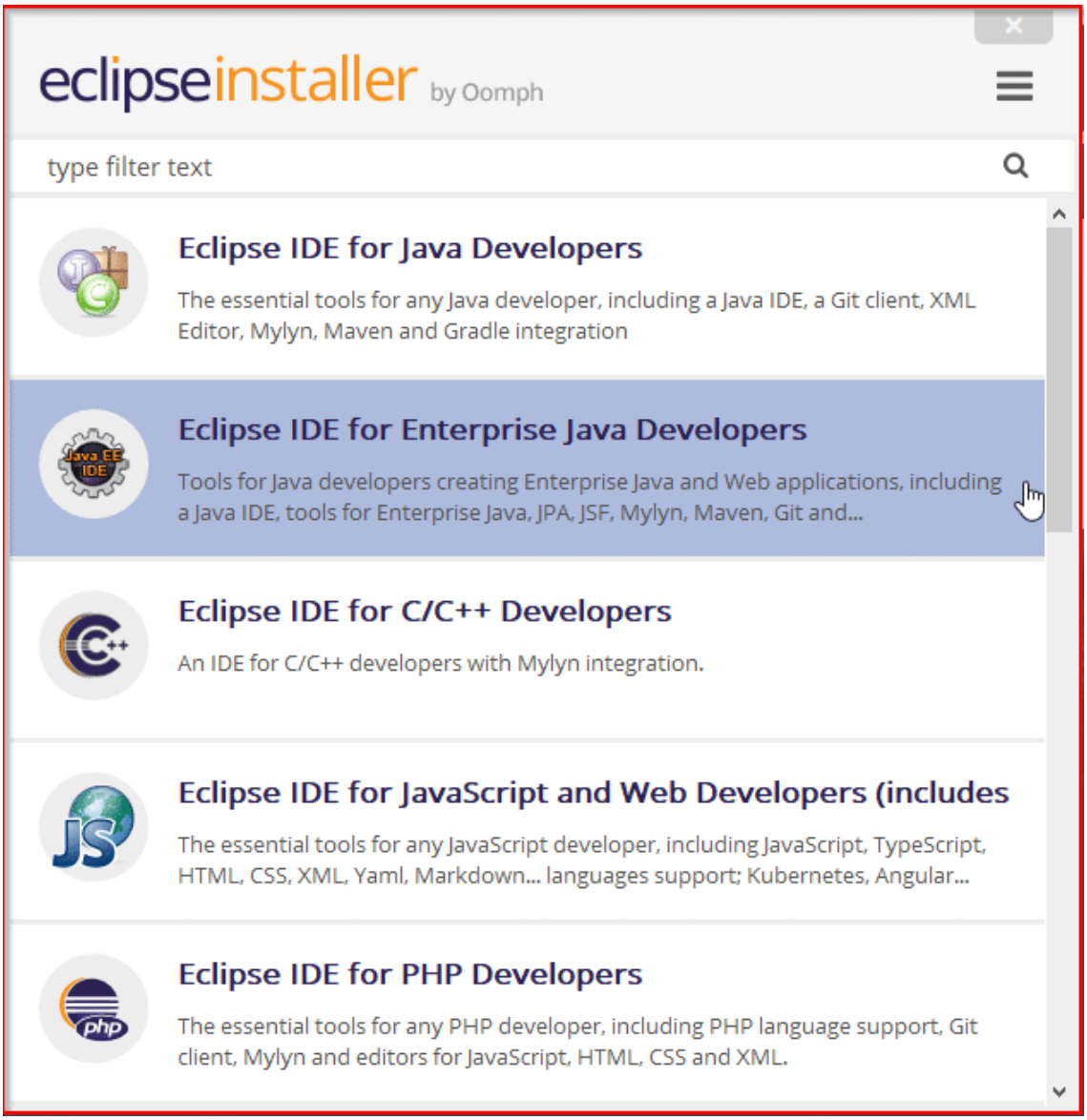
Step 1: Go to the Eclipse download page (the recent one is Eclipse Installer 2019-06 R), with this link: https://www.eclipse.org/downloads/



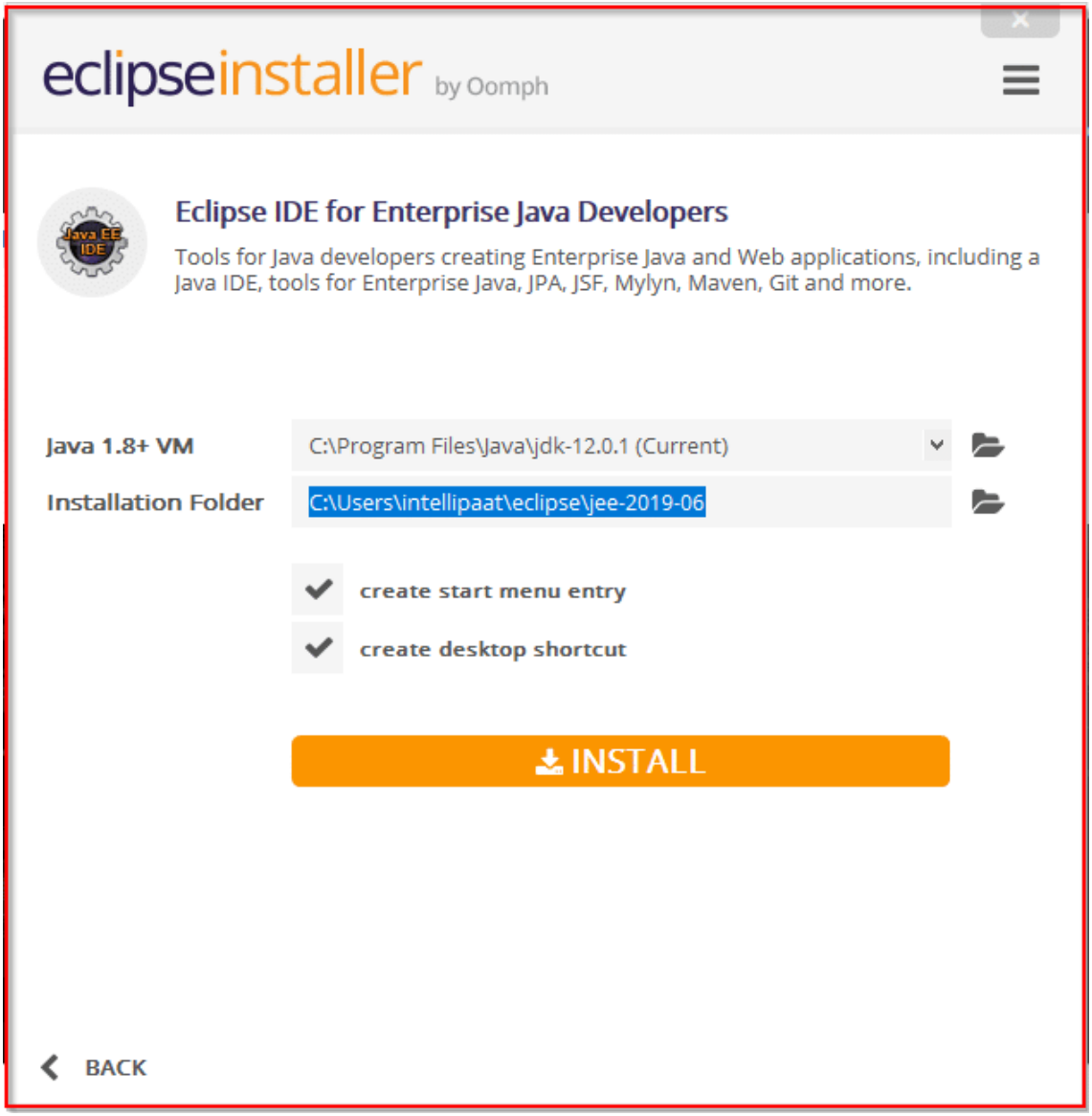
Step 2: Click on **Download 64 bit**. You would land on a page as shown below:



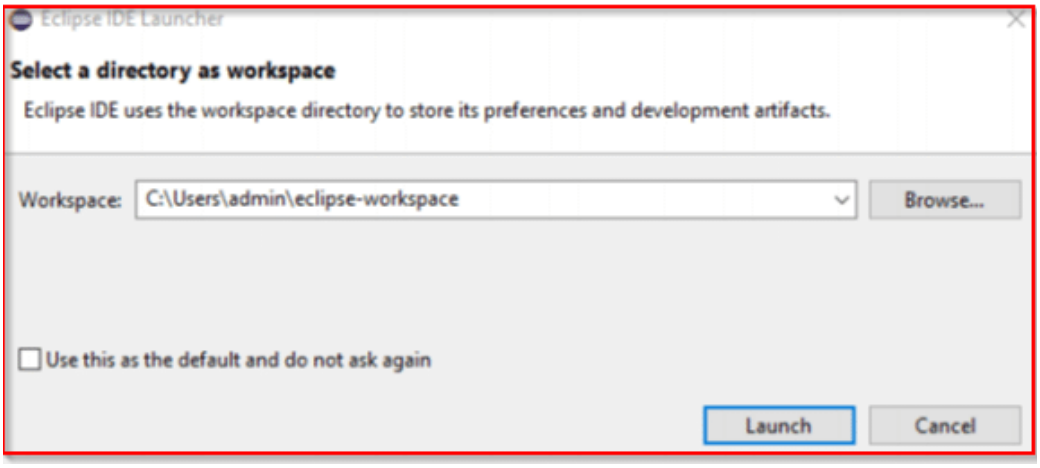
Step 3: Once the download is finished, launch the installer
Step 4: Click on **Eclipse IDE for Enterprise Java Developers**



Step 5: Once done, click on **INSTALL**



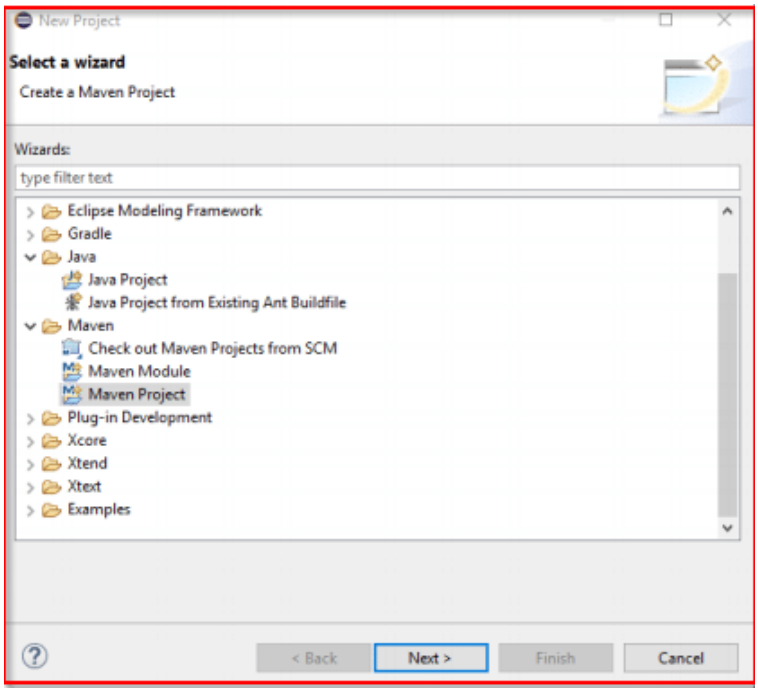
Step 6: Then, click on **Launch**



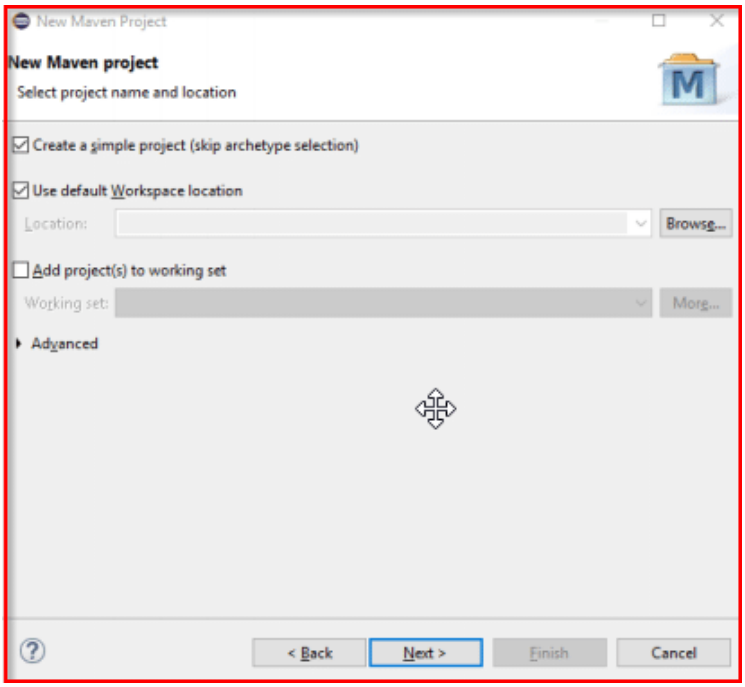
Now that you have successfully set up Java and Eclipse in your environment, let's perform a Selenium Test Case using Maven.

Performing Selenium Test Case

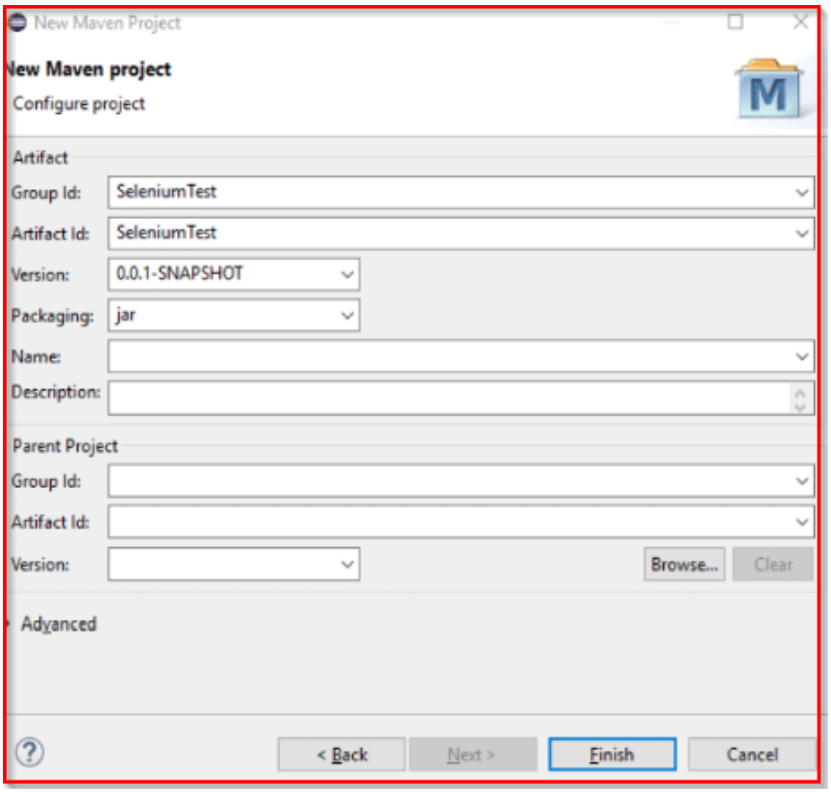
Step 1: Start a new **Maven Project**



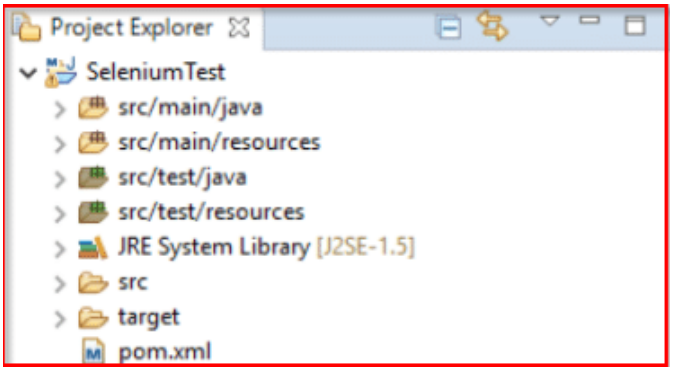
Step 2: Check the **Create a simple project** checkbox and click on **Next**



Step 3: Enter **Group Id** and **Artifact Id** and click on **Finish**
Group Id: Group Id is the Id of the project's group. Generally, it is unique among an organization.
Artifact Id: Artifact Id is the Id of the project. It specifies the name of the project.



Step 4: Now, your project would appear in the **Project Explorer** section as shown below:

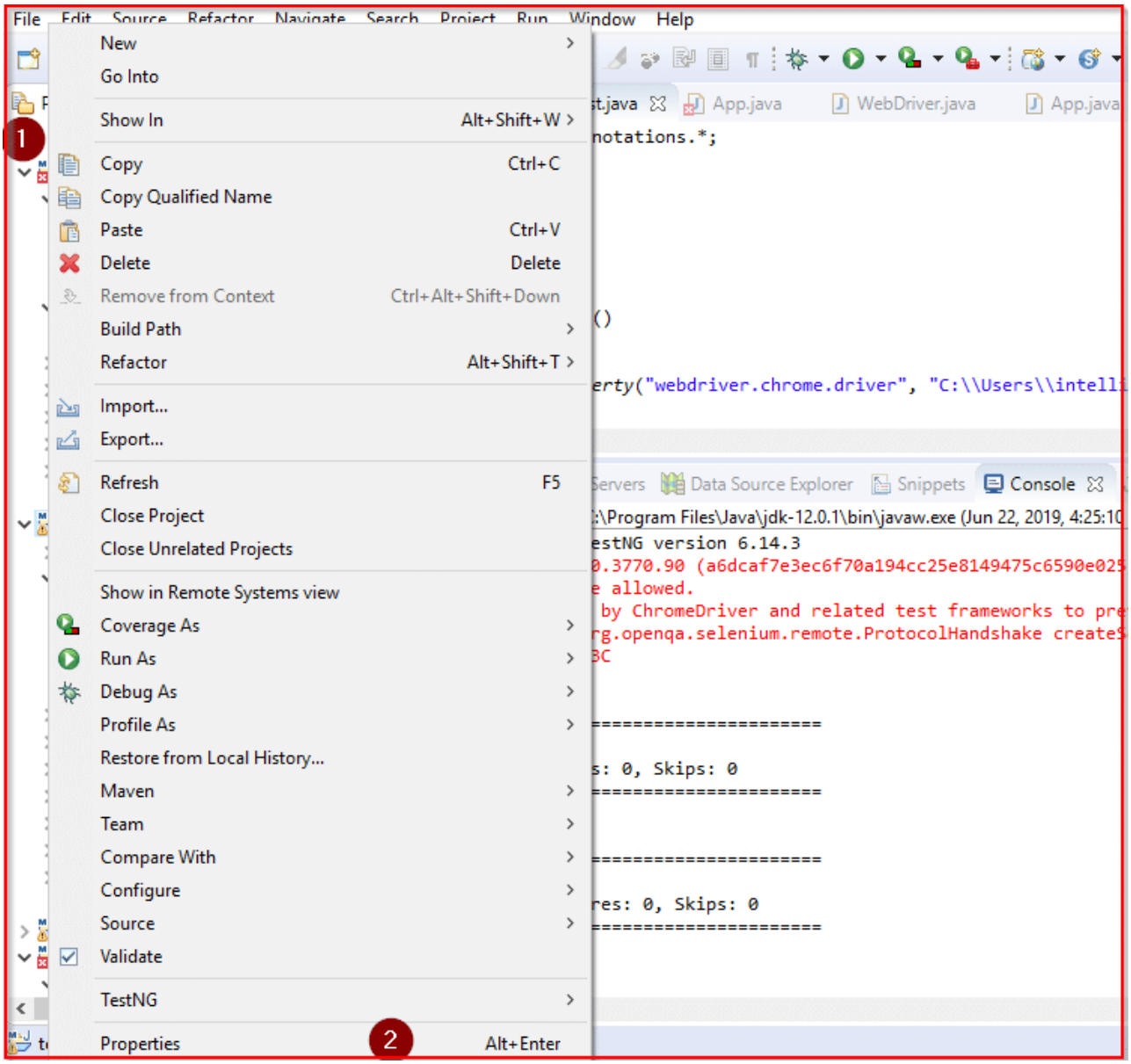


Before moving on to the scripting part, you need to configure the Maven Dependencies to perform Selenium Test Case in Eclipse. You will be adding the Maven Dependencies to the pom.xml file under the target folder.
Step 5: Now, go ahead and add the below dependencies to the pom.xml file
Note: Copy the code from below and paste it after the junit dependency which will be already present in your .xml file

```
<dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <version>3.141.59</version>
    </dependency>

    <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-server</artifactId>
            <version>3.141.59</version>
    </dependency>

    <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-api</artifactId>
            <version>3.141.59</version>
    </dependency>
    <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-chrome-
driver</artifactId>
            <version>3.141.59</version>
    </dependency>
```

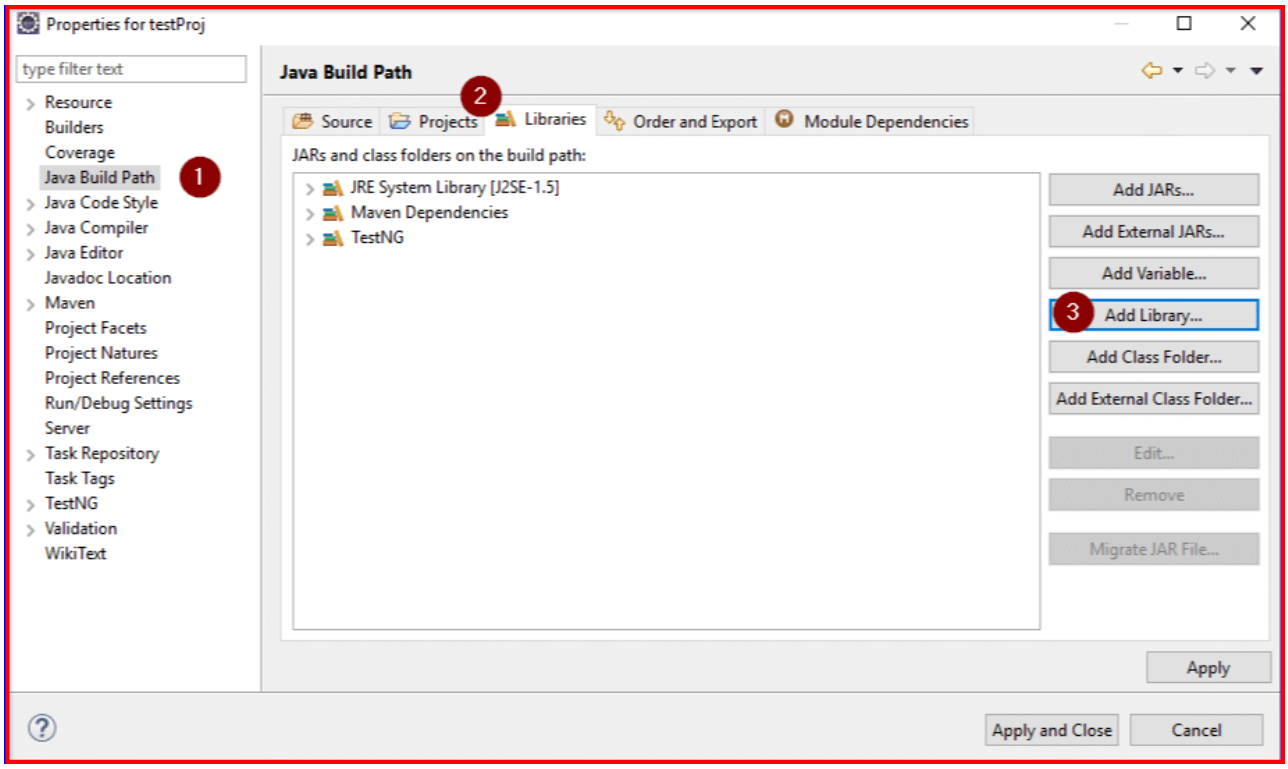
Step 6:

1. Right-click on the file name
2. Click on **Properties**

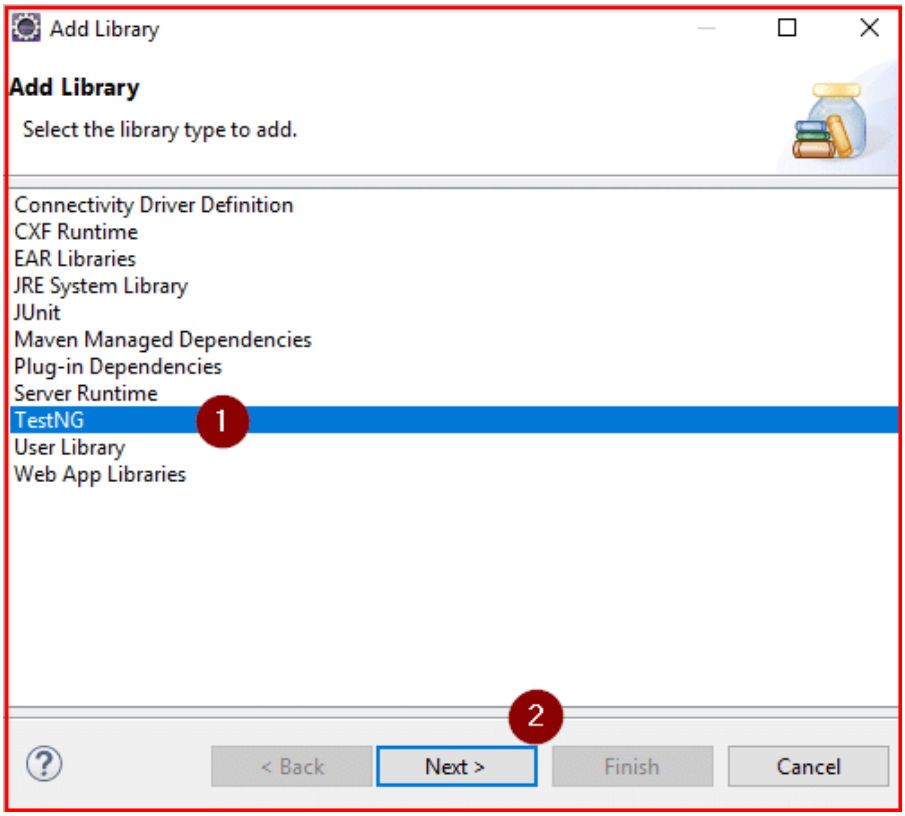


Step 7:

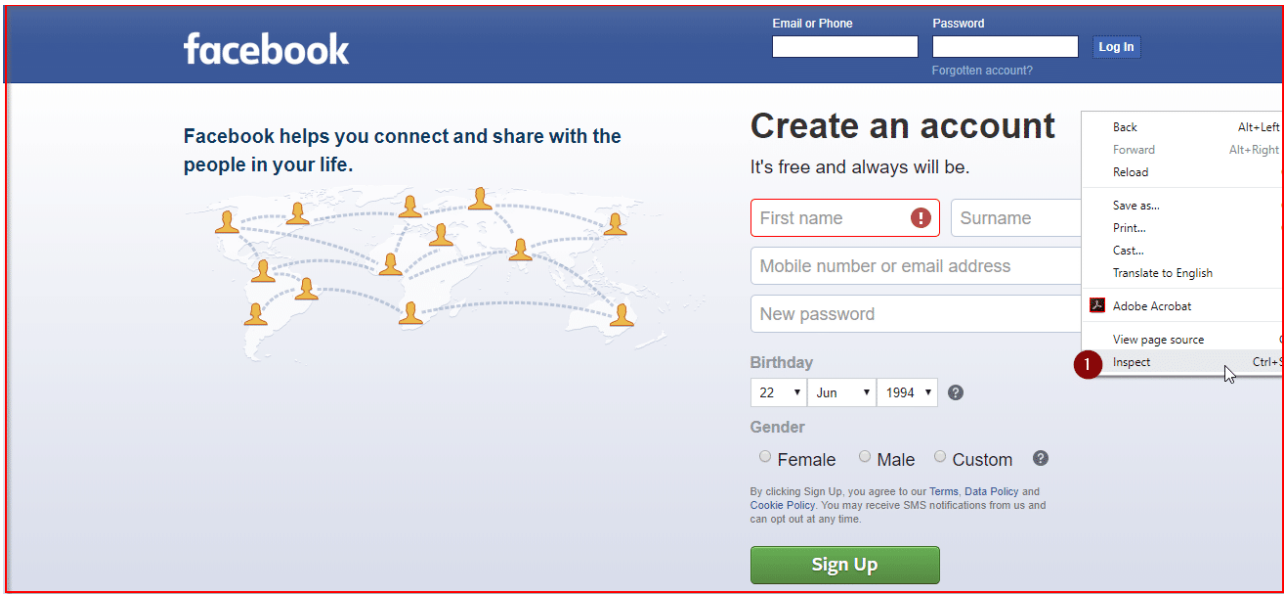
1. Select **Java Build Path**
2. Click on **Libraries**
3. Click on **Add Library**



Step 8:
1. Click on **TestNG**
2. Press **Next** and then **Finish**

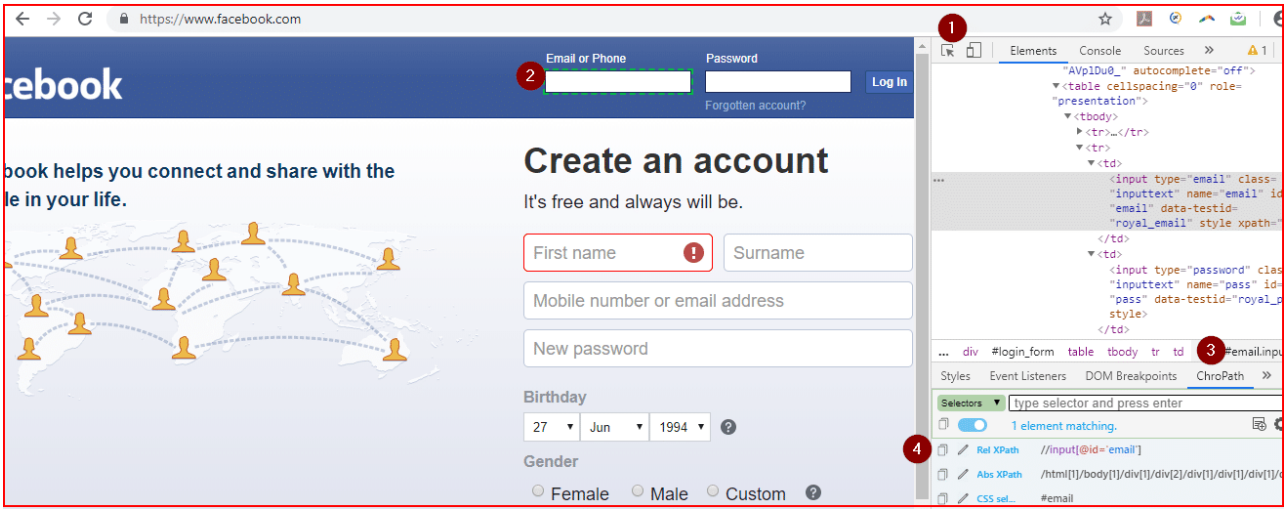


Step 9: Click on the **Apply and Close** button
Step 10: Go to **www.facebook.com**, right-click on the page, and click on **Inspect**



Step 11: Follow the process shown below and copy the xpath of **email:**

1. You will use this **arrow** to select an element in the page to inspect it
2. Select the **email** web-element, which will point to the section of this web-element's html code
3. Click on **Chropath**, this helps you in getting Xpaths and CSS selectors for web elements of a web page, uniquely
4. Copy the RelXpath. XPath is defined as XML path. It is a syntax or language **for locating** any **element** on **the net** page **using** XML path expression

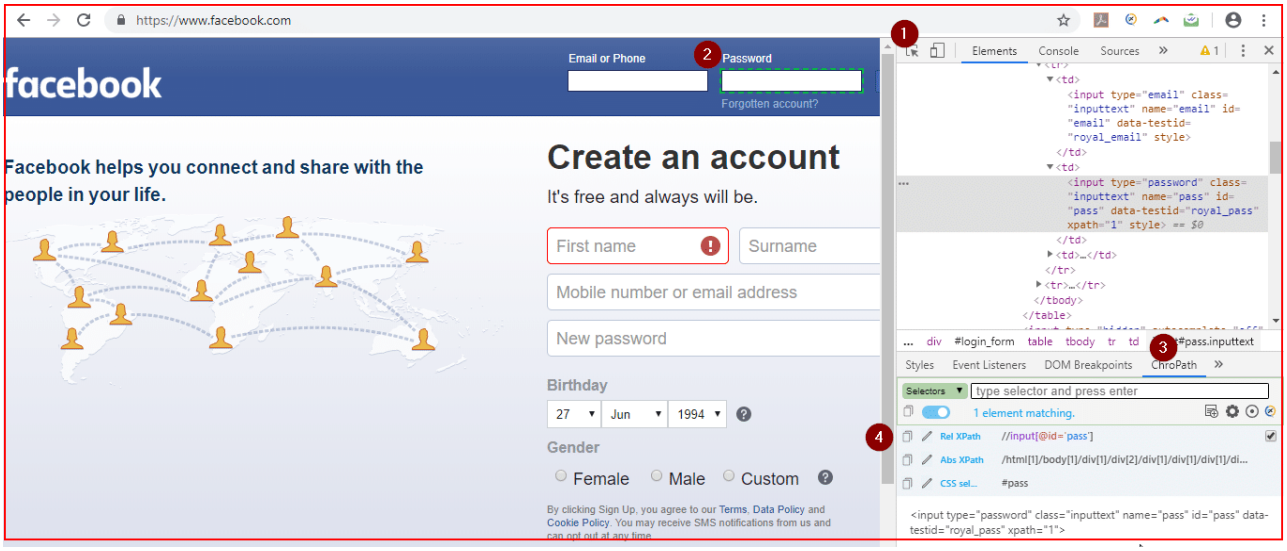


Step 12:

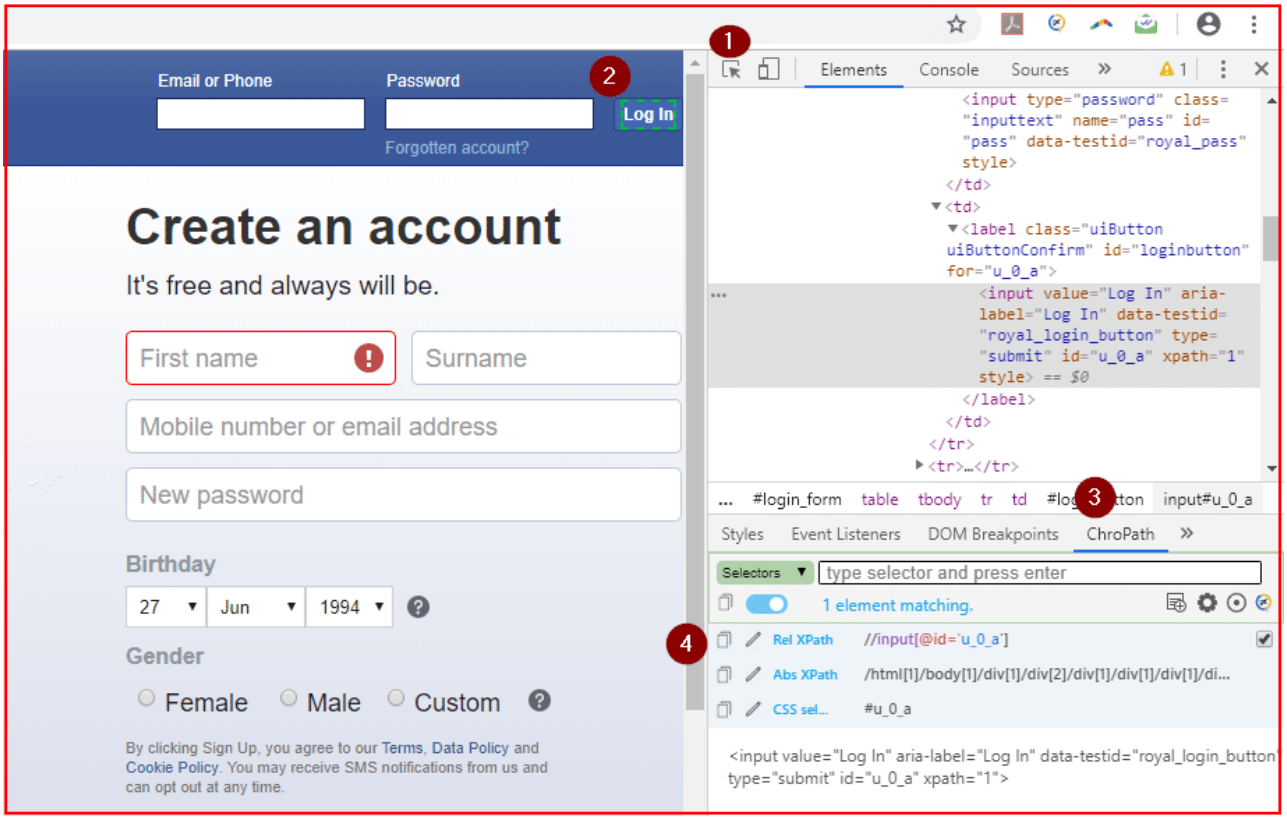
1. Paste the xpath in the xpath expression (i.e., key)
2. Enter the value (email ID)

```
package jar;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.*;

@Test
public class App

{
 WebDriver driver;

    public void test()
    {
    System.setProperty("webdriver.chrome.driver", "C:\\Users\\intellipaat\\Desktop\\chromedriver.exe");


      driver = new ChromeDriver();
      //The website name you want to visit
      driver.get("http://www.facebook.com");
      driver.findElement(By.xpath("//input[@id='email']")).sendKeys("helloworld@gmail.com");


    }
}
```

Step 13: Do the same for **password** xpath



Step 14: Do the same for **login** xpath as well



Step 15: Your code would look like this:

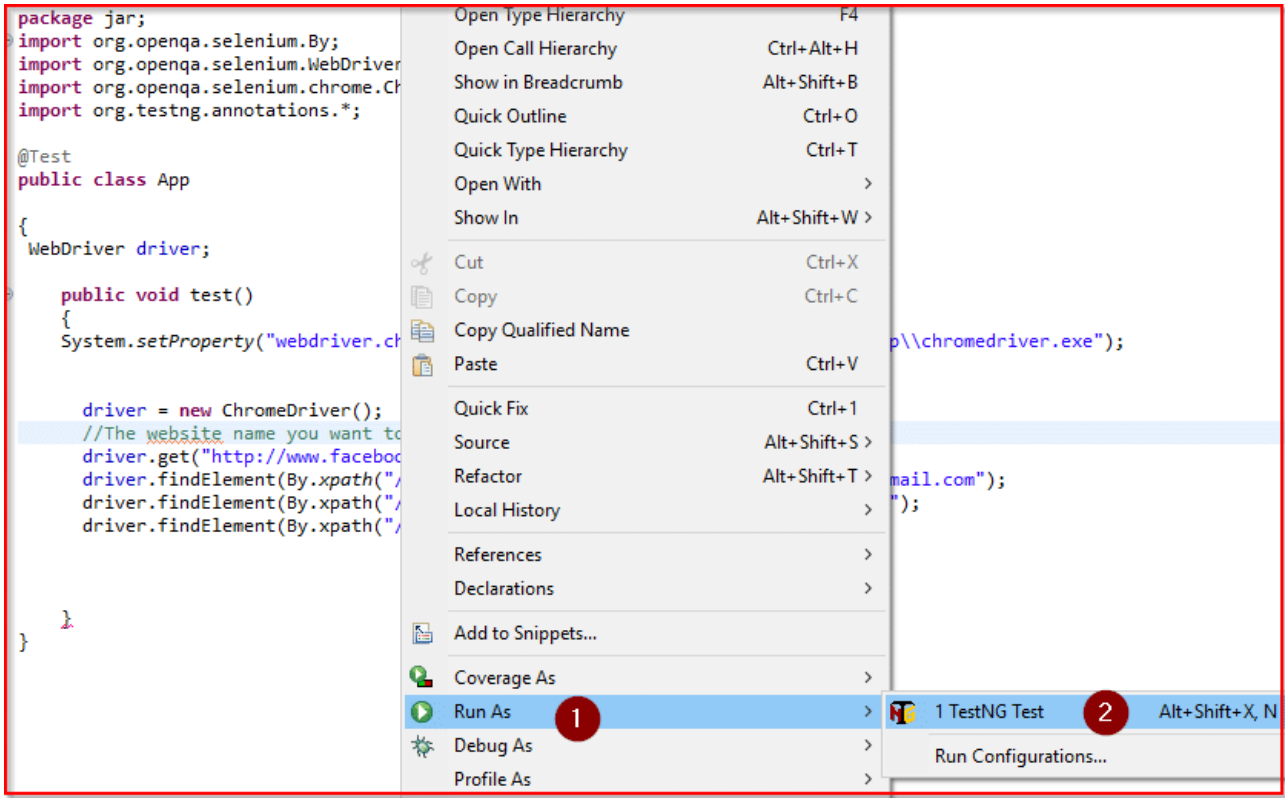
```
@Test
public class App

{
 WebDriver driver;

            public void test()
            {

System.setProperty("webdriver.chrome.driver","C:\\Users\\intellipaat\\Desktop\\chromedriver.e
xe");


                         driver = new ChromeDriver();
                        //The website name you want to visit
                        driver.get("http://www.facebook.com");
      driver.findElement(By.xpath("//input[@id='email']")).sendKeys("[email protected]");
      driver.findElement(By.xpath("//input[@id='pass']")).sendKeys("helloworld123");
                        driver.findElement(By.xpath("//input[@id='u_0_2']")).click();
              }
}
```

Note: Here, you must provide a valid email ID and password. In this example, dummy ID and password would be used.

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.*;

@Test
public class App

{
 WebDriver driver;

    public void test()
    {
    System.setProperty("webdriver.chrome.driver", "C:\\Users\\intellipaat\\Desktop\\chromedriver.exe");


      driver = new ChromeDriver();
      //The website name you want to visit
      driver.get("http://www.facebook.com");
      driver.findElement(By.xpath("//input[@id='email']")).sendKeys("helloworld@gmail.com");
      driver.findElement(By.xpath("//input[@id='pass']")).sendKeys("helloworld123");
      driver.findElement(By.xpath("//input[@id='u_0_2']")).click();



    }
}
```

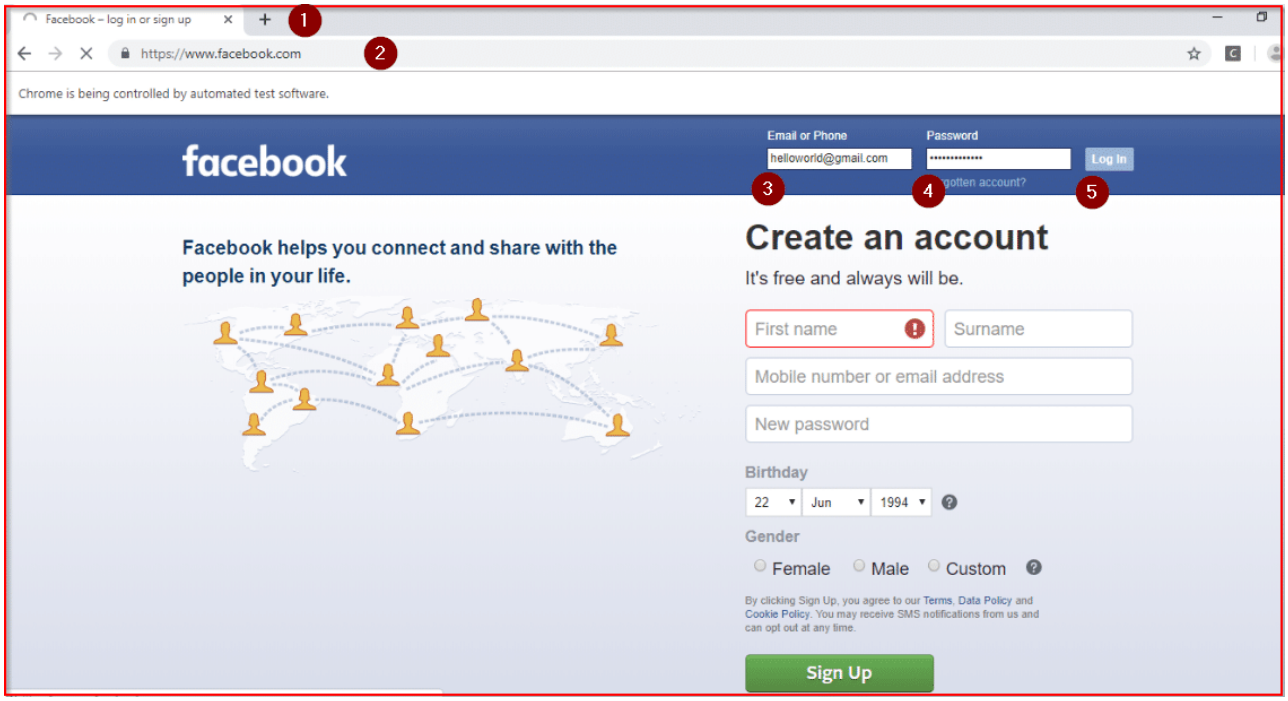
Step 16:

1. Right-click on the screen and scroll down to **Run As**
2. Click on **1 TestNG Test**

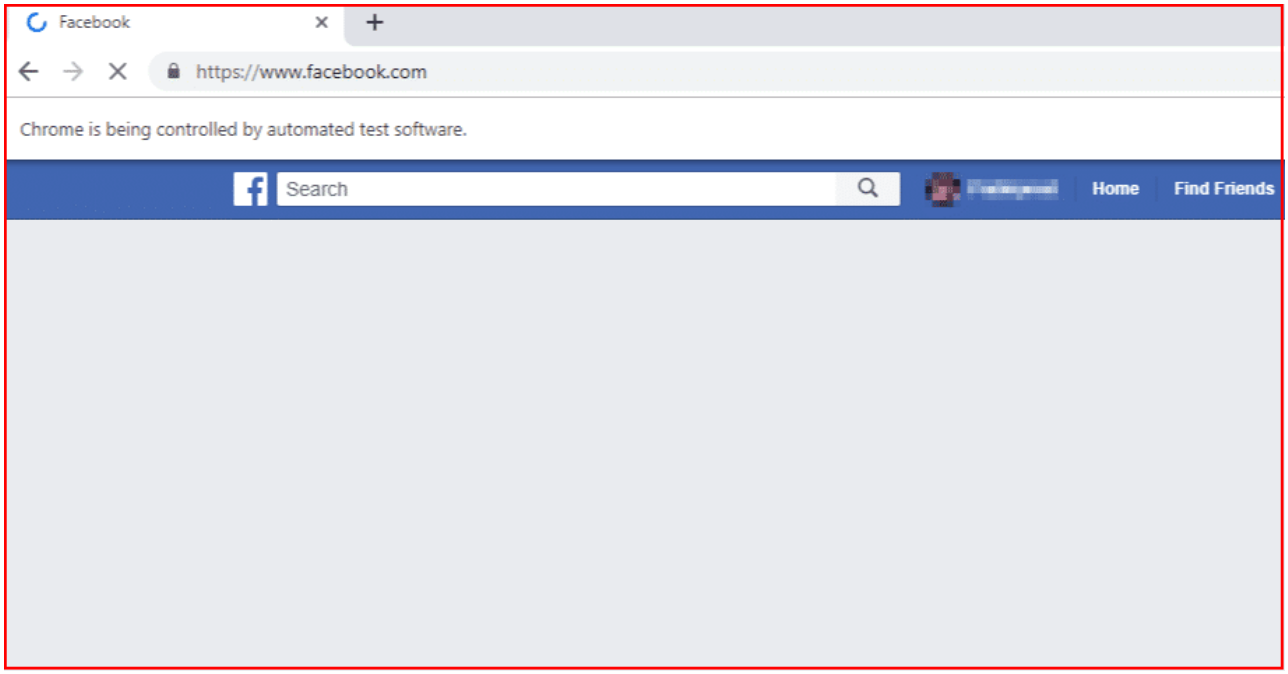


Step 17:

1. Once done, the Chrome browser window will pop up
2. It will search for facebook.com
3. Then, it will automatically type the email
4. It will automatically type the password as well
5. It will then click on **Log in** and navigate you to your FB homepage



Step 18: You'll be navigated to your FB account



Step 19: Go to **Eclipse IDE** again and see the console. You would see the Test report with the results

```
Markers  Properties  Servers  Data Source Explorer  Snippets  Console  JUnit  Results of running class AppTest
<terminated> AppTest [TestNG] C:\Program Files\Java\jdk-12.0.1\bin\javaw.exe (Jun 22, 2019, 4:25:10 PM)
[RemoteTestNG] detected TestNG version 6.14.3
Starting ChromeDriver 75.0.3770.90 (a6dcaf7e3ec6f70a194cc25e8149475c6590e025-refs/branch-heads/3770@{#1003}) on port 4079
Only local connections are allowed.
Please protect ports used by ChromeDriver and related test frameworks to prevent access by malicious code.
Jun 22, 2019 4:25:13 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
PASSED: test

===============================================
    Default test
    Tests run: 1, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 1, Failures: 0, Skips: 0
```

That's it! You have successfully performed a Selenium Test Case.

Creating Automated Tests

There are three steps to execute automated tests:

- Find an element on the web browser
- Perform action on that element
- Test and create a test report with the results

Finding an Element on the Web Browser

An element on a web browser can be found using:

- ID
- Name
- Class Name
- Tag Name

Performing an Action on That Element

The next step is performing an action. For that you can try the following options:

- Click(): Used to click on a web element
- sendKey(): Used to take values into text boxes
- Clear(): Used to clear textboxes of its current value
- Submit(): WebDriver will automatically trigger the submit function of the web application where that element belongs to

Testing and Reporting

Report generation is extremely vital once you do the Automation Testing, likewise as for Manual Testing. By looking at the result, you can easily identify how many test cases are passed, failed, and skipped. By viewing the report, you'll come back to understand what the status of the project is. Selenium Internet Driver is employed for automating the web-application, however, it will not generate any reports. TestNG will generate the default report.

Selenium Career Opportunities

As numerous companies have started using web applications, the need for Selenium is shooting up. It has grown tremendously and has become one of the most integral parts of various businesses. When it comes to web testing tools, Selenium has surely become one of the best in the industry to help you with automation testing services.
There are various **Selenium career opportunities** for a tester. Below mentioned are some of the popular job roles you can consider to work as a Selenium WebDriver professional:

- Automation Test Lead
- Senior Test Engineer
- Quality Engineer
- Selenium Automation Analyst
- QA Engineer

The most interesting part here is that the industry has experienced a humongous surge in creating job opportunities with very exciting packages. So, you must take the assistance of experts and get yourself qualified enough to make the best of these opportunities.

Interested? Check out Intellipaat's **Selenium Training – WebDriver Course** and enroll now!

Frequently Asked Questions

What is a Selenium test tool and how does it work?

Selenium, a free automated suite of software for testing web applications offers a playback/record tool for writing tests without having to learn the test scripting language. When learning Selenium, you can automate your web application for testing.

How Selenium is better than other testing tools?

Selenium is open-source software that anyone can download and use for free. It is used to automate web browser and supports a variety of programming languages to write test scripts and supports cross-browser testing. Moreover, it is easily integrable with other tools and supports parallel & distributed testing and has no dependency on GUI-based systems.

Why learn Selenium?

The industry's demand for test automation continues to grow and Selenium is the world's most popular test automation tool. The average salary of Selenium Automation Engineer is $81,405 (PayScale) in the US and INR 5,49,436 (Glassdoor) in India. Thus, as a professional looking to make a mark in the industry, you must learn Selenium.

Is it easy to learn Selenium?

Yes, to learn Selenium, the basic knowledge of OOPS concept is enough. This Selenium tutorial will provide you with valuable insights into learning Selenium, its installation, features and more.

Although the tools help, they won't give you any knowledge. So, if you want to learn selenium start with learning test automation first.

Can I learn Selenium without knowing Java?

You don't have to know all the features of Java because it is not required for automated testing of Selenium. Even as a non-programmer, you can learn a lot from this Selenium tutorial. However, for the most commonly used Selenium web drivers, you need basic Java concepts like OOP.

What is Selenium used for?

Selenium is mainly used to test web applications. It also provides a domain-specific language for writing test cases in a common programming language, such as Scala, Java, C#, Ruby, and more. Selenium automated test tools are used by most companies for their products.

What are the different components of Selenium?

Selenium has four components, including, Selenium Remote Control (RC), Selenium Integrated Development Environment (IDE), WebDriver, and Selenium Grid. The last two are the most famous. Each has its own unique features, including, support for multiple browsers, parallel testing, and more.

Which programming language is best for Selenium?

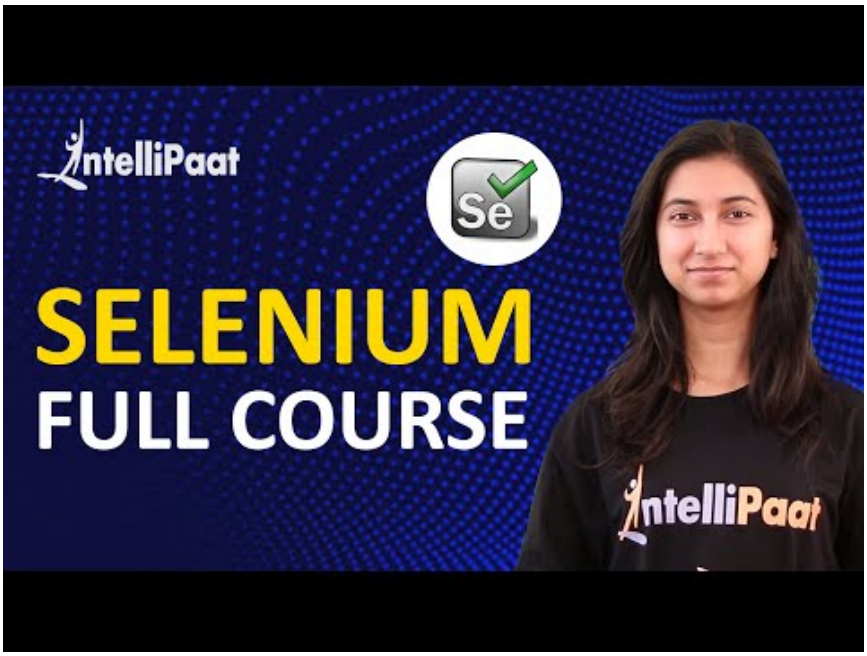
For learners who are not comfortable with programming, can use Python for its script friendly feature, or Ruby. This Selenium tutorial will improve your learning curve and help you understand how to write Selenium tests with fewer codes.

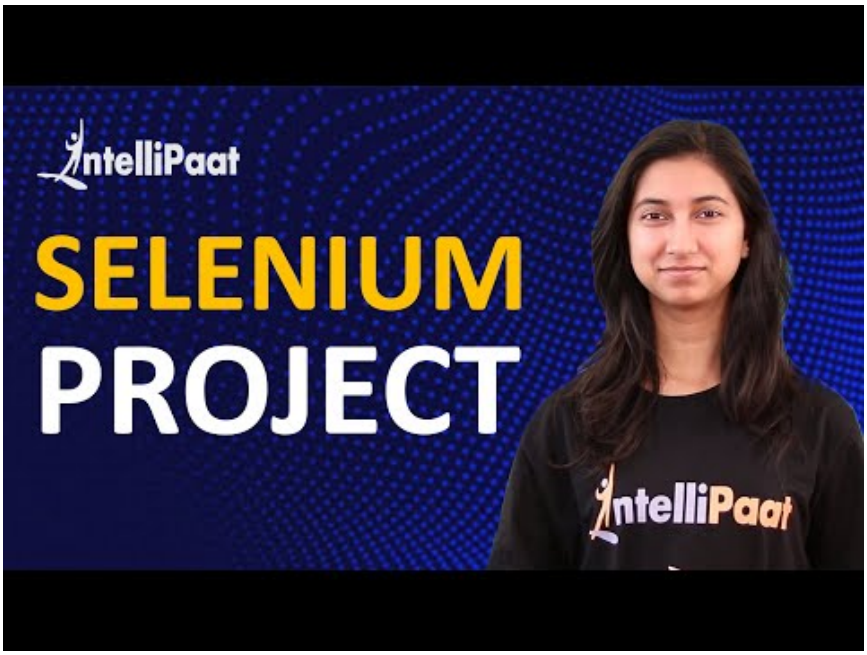
Next

Recommended Videos


IntelliPaat
SELENIUM
TUTORIAL
FOR BEGINNERS
IntelliPaat


IntelliPaat
Selenium automation
Testing
Redefining Learning
Subscribe


IntelliPaat
SELENIUM
FULL COURSE
IntelliPaat



Course Schedule

Name	Date	
Test Architect	2021-01-30 2021-01-31 (Sat-Sun) Weekend batch	View Details
Test Architect	2021-02-06 2021-02-07 (Sat-Sun) Weekend batch	View Details
Test Architect	2021-02-13 2021-02-14 (Sat-Sun) Weekend batch	View Details

3 thoughts on “Selenium Tutorial for Beginners using Python”

Leave a Reply

Your email address will not be published. Required fields are marked *

What is Selenium?

Selenium Overview

Selenium is a free (open-source) automatic testing suite for web applications across completely different browsers and platforms. It is quite the same as HP QuickTest Pro (QTP, currently UFT); however, Selenium focuses on automating web-based applications. Testing done using Selenium tool is usually referred to as Selenium Testing. Remember that, testing of only web applications is possible with Selenium. We can test neither desktop applications nor mobile applications using Selenium.

Watch this Selenium Tutorial:

As part of this tutorial, you will learn the following topics:

There is no need of feeling disappointed due to this because other software has got you covered as there are many tools for testing desktop and mobile applications like IBM's RFI, HP's QPI, Appium, etc. But, the aim of this tutorial is to understand the testing of dynamic web applications and why Selenium is the best for this purpose.

To learn more, visit our Selenium Community and get your doubts clarified!

Since Selenium is open-source, there is no licensing cost involved, which is a significant benefit over other testing tools. Other reasons behind Selenium's ever-growing popularity are as follows:

- Test scripts are often written in any of these programming languages: Java, Python, C#, PHP, Ruby, Perl, and .Net.
- Tests can be carried out in any of these operating systems: Windows, Mac, or Linux.
- Tests can be carried out using any of these browsers: Mozilla Firefox, Internet Explorer, Google Chrome, Safari, or Opera.
- It can be integrated with tools like TestNG and JUnit for managing test cases and generating reports.
- It is integrated with Maven, Jenkins, and Docker to achieve continuous testing.

Now, let us talk about Selenium in particular, and let us see where Selenium stands in the market.

Watch this Selenium Tutorial

Selenium vs QTP

A comparison based on the performance factor of Selenium with another popular tool, QTP, is shown below.

Go through the Selenium Course in London to get clear understanding of Selenium.

A Comparison Between Selenium and QTP (Now UFT)

QuickTest Professional (QTP) is a proprietary automated testing tool previously owned by the company, Mercury Interactive before it was acquired by Hewlett Packard in 2006.

Learn more about Selenium in this Selenium training in New York to get ahead in your career!

Selenium Tool Suite has several benefits over QTP as elaborated below:

Advantages of Selenium over QTP

Selenium:

- Is open-source, which means that it is free to use
- Is Highly extensible
- Runs tests across different browsers
- Supports various operating systems
- Supports mobile devices
- Executes tests while the browser is minimized
- Executes tests in parallel

Interested in learning Selenium? Learn more from this Selenium Training in Sydney!

QTP:

- Is commercial
- Has limited add-ons
- Only be used in Windows
- Supports mobile apps test automation (iOS and Android) using HP solution called HP Mobile Center
- Needs to have the application under test to be visible on the desktop
- Executes only in parallel but using Quality Center, which is again a paid product

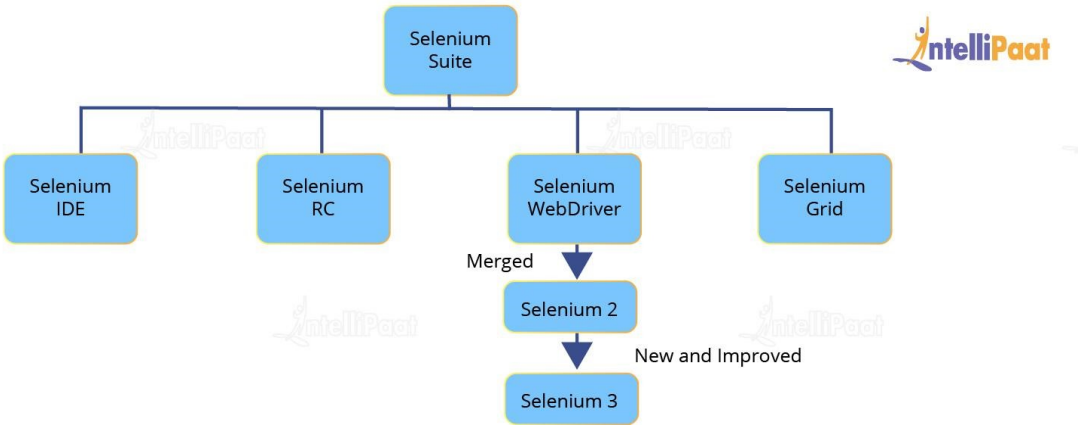
It is pretty clear from the above comparison why Selenium is the most popular tool. But, there are several other nuances in Selenium, and you need to understand which one is the most applicable Selenium tool for your requirements.

You would want to check out these top Selenium Interview Questions to ace your next Selenium testing interview!

Selenium Tools

Selenium isn't simply one tool but a collection of software, each catering to different testing needs of an organization. It has four components:

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



Let's now discuss each component in detail.

- Selenium IDE: IDE stands for Integrated Development Environment, which is a plug-in of the browser, Firefox. This is the simplest framework and therefore requires developers to switch to Selenium RC for advanced test cases.
- Selenium RC: RC stands for remote control and it permits developers to code in the language of their preference. Selenium RC supports multiple programming languages such as Java, C#, Python, etc.
- Selenium WebDriver: Selenium WebDriver automates and controls activities undertaken by a web browser. It does not typically depend on JavaScript but communicates with the browser to control its actions. Like RC, it additionally supports programming languages such as Java, C#, Python, Ruby, etc.
- Selenium Grid: Selenium Grid is employed to execute parallel tests along with Selenium RC across multiple browsers.

Become a master of Selenium by going through this online Selenium Course in Toronto!

Watch this Automation Testing vs Manual Testing Tutorial:

How to Choose the Right Selenium Tool for Your Needs

Project to project, the testing requirements will be different, and Selenium, an open-source automated testing suite, has got you covered on this with its collection of different tools.

Selenium IDE

Selenium IDE (Integrated Development Environment) is the tool you use to develop your Selenium test cases. When you create your first IDE script, it may include the following concepts:

- Selenese commands such as clickAndWait, assert, verify, type, open, etc.
- Locators like ID, Name, XPath, CSS Selector, etc.
- Executing customized JavaScript code using RunScript
- Exporting test cases in various formats like .cs (C# source code),.java (Java source code),.py (Python source code),.rb (Ruby source code)
- To create tests with very little or no previous knowledge in programming
- To create simple test cases and test suites that you can export later to RC or WebDriver
- To test an online application only against Firefox and Chrome

Learn more about Selenium from this insightful blog on How to Test Web Applications Using Selenium!

Selenium RC

- To design a test using a more expressive language than Selenese
- To run your test against different browsers (except HtmlUnit) on different operating systems
- To deploy your tests across multiple environments using Selenium Grid
- To test your application against a new browser that supports JavaScript
- To test web applications with complex AJAX-based scenarios

Selenium WebDriver

- To use a particular programing language in designing your test suit
- To test applications that are rich in AJAX-based functionality
- To execute tests on the HtmlUnit browser
- To create customized test results

Selenium Grid

- To run your Selenium RC scripts in multiple browsers and operating systems at the same time
- To run a large test suite that must complete within the soonest time possible

In this tutorial you have learned about various Selenium tools, advantages of Selenium over QTP and how to choose the right Selenium tool for your needs.

If you are interested to learn Selenium at a much deeper level and want to become a professional in the testing domain, ***check out Intellipaat's Selenium Training Certification Course*.**

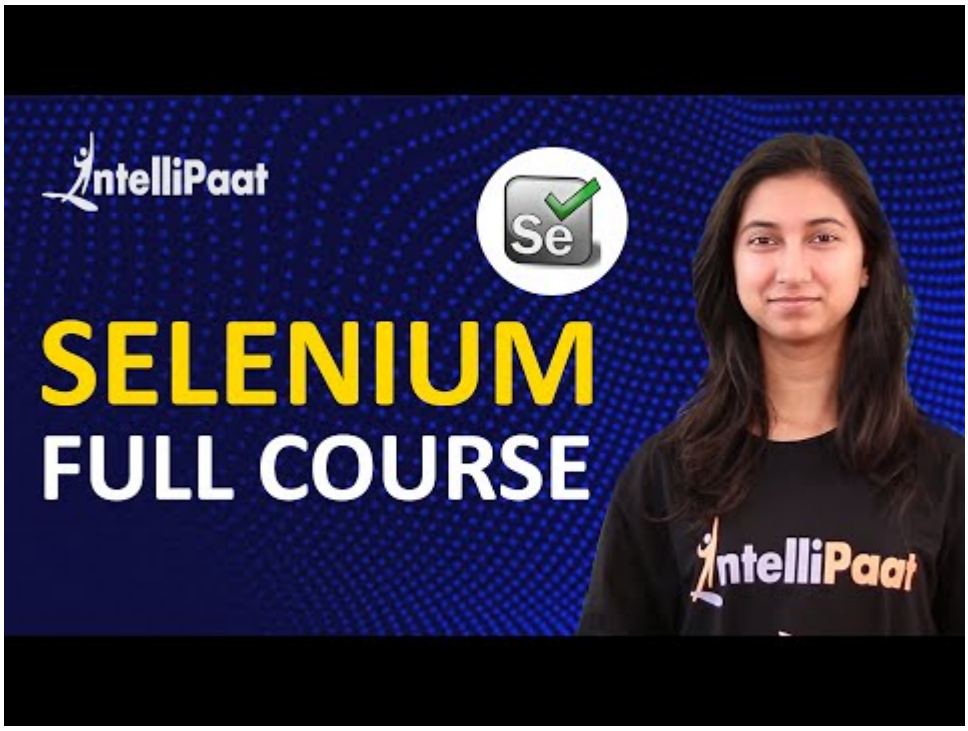
For a more detailed explanation of Selenium WebDriver, browser elements, how to locate browser elements present on a web application, and setting up Selenium with Maven and TestNG on Eclipse, ***check out this Selenium Tutorial!***

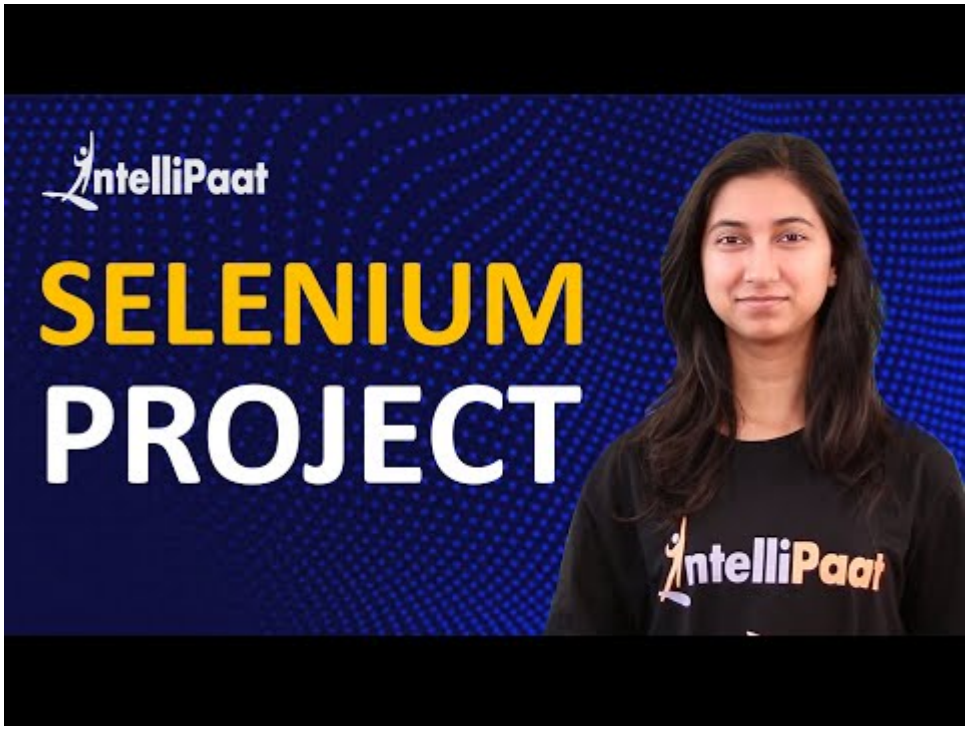
Previous Next

Recommended Videos


IntelliPaat
SELENIUM
TUTORIAL
FOR BEGINNERS
IntelliPaat


IntelliPaat
Selenium automation
Testing
Redefining Learning
Intellipaat
Subscribe





Course Schedule

Name	Date	
Test Architect	2021-01-30 2021-01-31 (Sat-Sun) Weekend batch	View Details
Test Architect	2021-02-06 2021-02-07 (Sat-Sun) Weekend batch	View Details

Test Architect	2021-02-13 2021-02-14 (Sat-Sun) Weekend batch	View Details

Leave a Reply

Your email address will not be published. Required fields are marked *

Installation - Selenium Tutorial

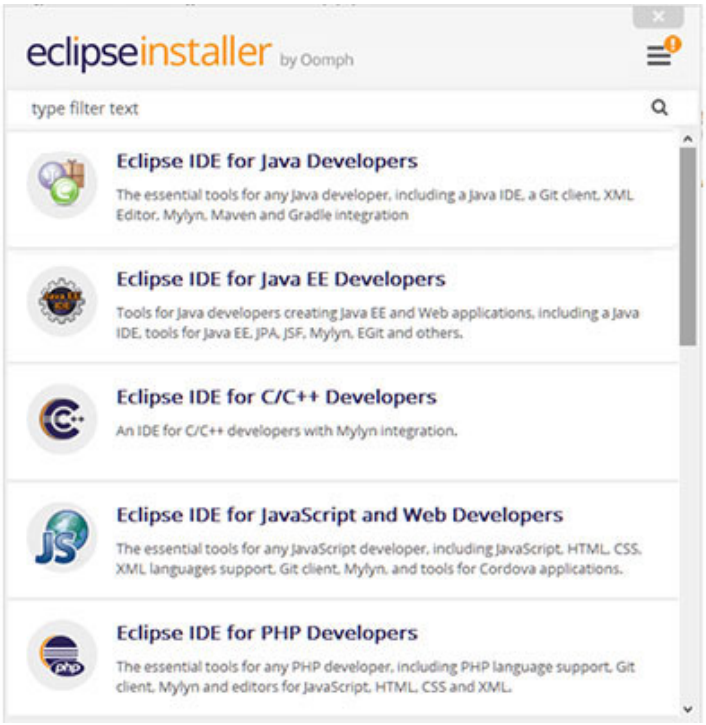
Updated on January 8, 2021 |
This section of the Selenium tutorial will help you understand the installation of Selenium, the step by step process of installing Selenium like installing Java, downloading Java Client Driver, creating a new Java project and so on.

Selenium Environment Setup

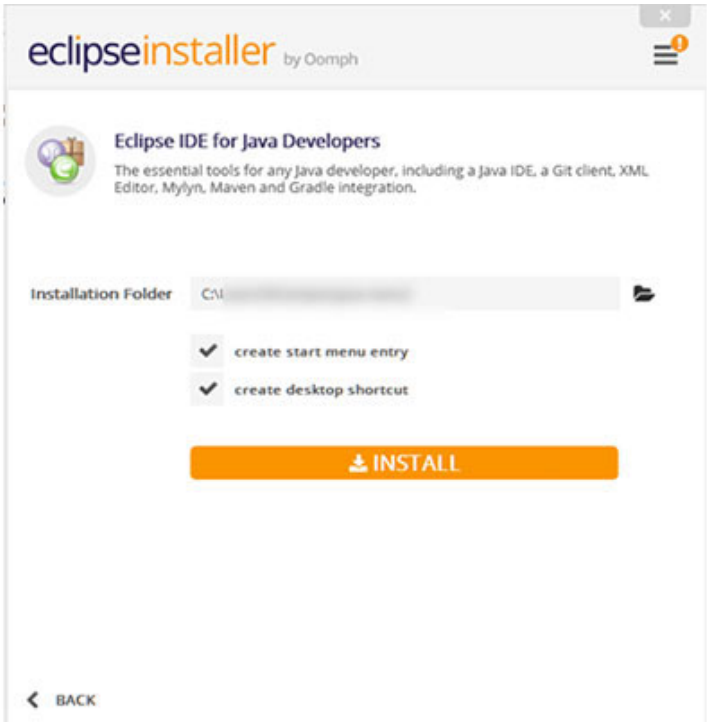
Installing Selenium requires going through following steps:
Step 1 : Install Java
Before moving further download and install Java Software Development Kit (JDK) in the first place. The location from where you can download Java is:
http://www.oracle.com/technetwork/java/javase/downloads/index.html

How to work on Selenium LIVE Project Tutorial

Step 2 : Install Eclipse IDE
Next, download and install Eclipse IDE for Java developers keeping in mind the 32 or 64-bit windows versions.
The location to download this exe file is:
http://www.eclipse.org/downloads/
The downloaded file will be with the name “eclipse-inst-win64”. Now double click on the file to install it on your system. A new window will appear with several options like this :



Click the first option named as Eclipse IDE for Java Developers. A new window will appear where you need to change the storage path to:
“c:\eclipse” and Install.

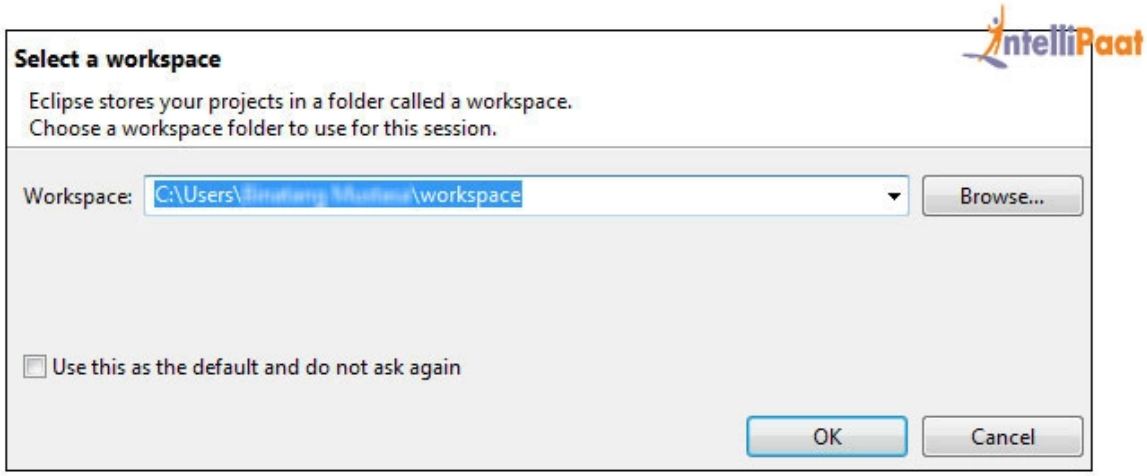


Step 3 : Download Selenium Java Client Driver
Now download Selenium Java Client Driver. The location to download this tool is http://docs.seleniumhq.org/download/
Here you will find the drivers for all the languages but install them for Java only. The drivers come in zipped file containing all the JAR files. There will be two JAR files, a lib folder and one change log file. Create a new folder and rename it as ‘Selenium’ in C drive

and extract the zip file to this file.

Step 4 : Configure Eclipse IDE

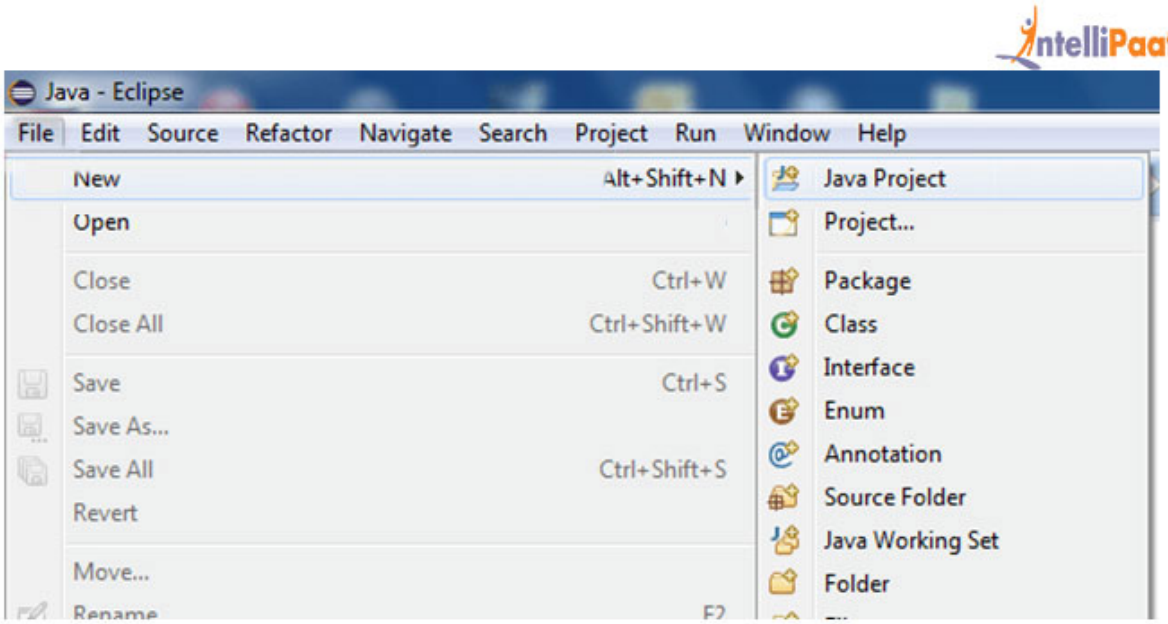
After completing the previous step a dialog box will open asking to choose a workspace. select the default location.



Or create a workspace in C drive and confirm it by pressing OK.

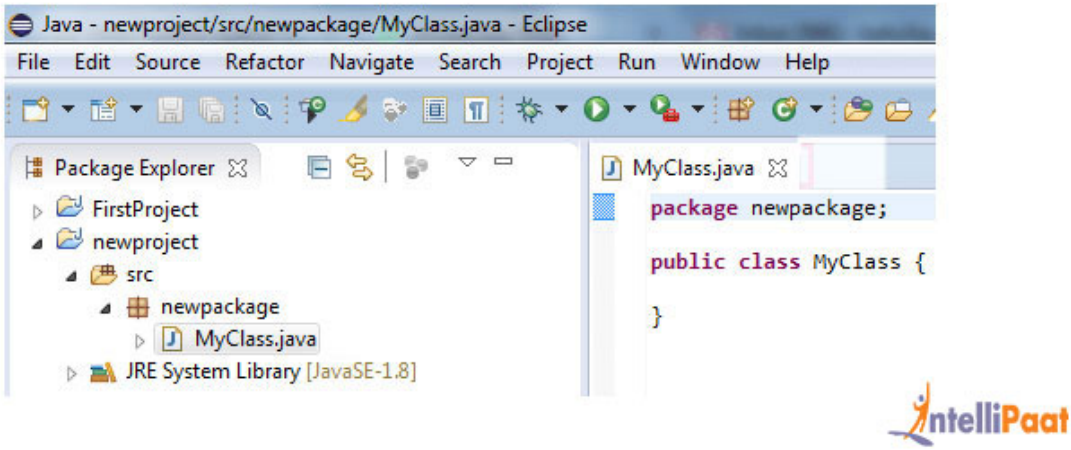
Step 5 : Create a new Java Project

Click on File menu, then go to New and select Java Project. Save the project as ‘newproject’.



Step 6 : Create a new Java Class

In order to create a new Java Class, click on File Menu and go to New and save it as ‘Myclass’.



Now right click on the newproject and go to select properties. There click on “Java Build Path”. Click on the ‘Libraries’ tab and then ‘Add External JARs’. ow select the JAR files you need from the new pop-up window and select ‘OK’. Add all the JAR files that are inside and outside the “libs” folder. Now click Ok to finish importing the library files.

Intellipaat offers you a world-class Online selenium Training delivered by trained experts! Enroll today!

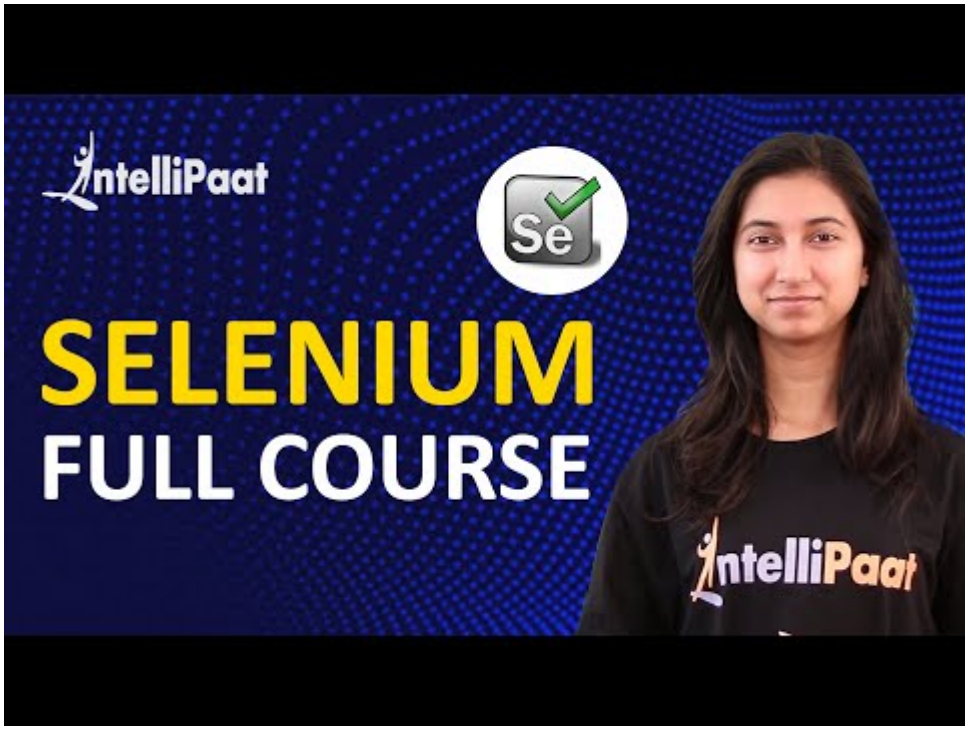
Previous Next

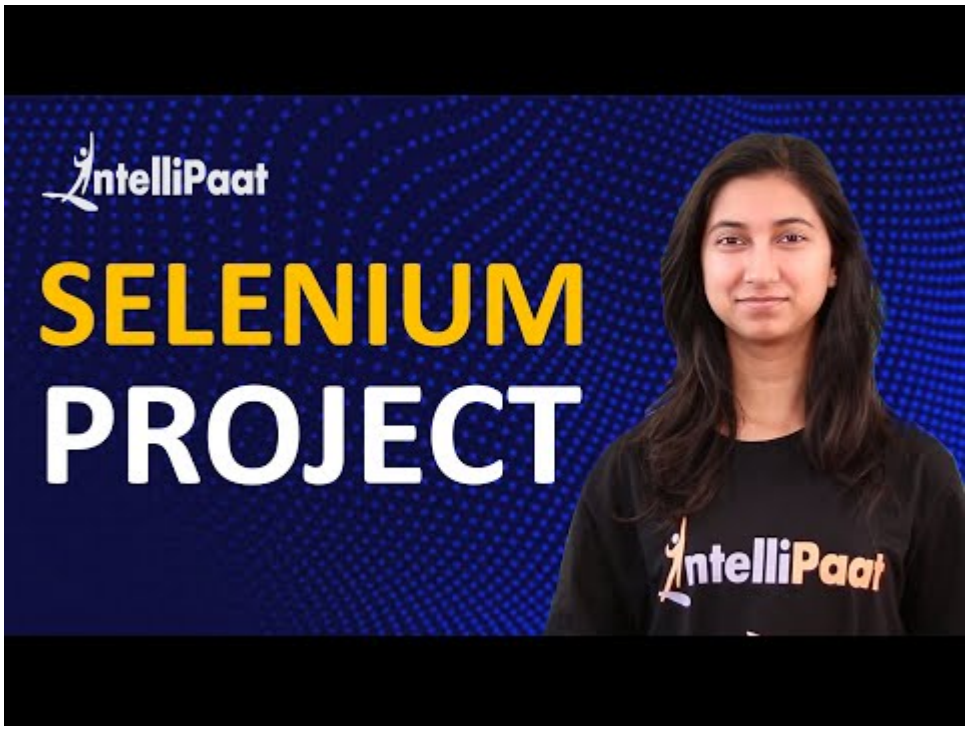
Recommended Videos


IntelliPaat
SELENIUM
TUTORIAL
MASTER TOP AUTOMATION
TESTING TOOL
IntelliPaat


IntelliPaat
SELENIUM
TUTORIAL
FOR BEGINNERS
IntelliPaat


IntelliPaat
Selenium automation
Testing
Redefining Learning
Intellipaat
Subscribe
Se


IntelliPaat
Se
SELENIUM
FULL COURSE
IntelliPaat



Course Schedule

Name	Date	
Test Architect	2021-01-30 2021-01-31 (Sat-Sun) Weekend batch	View Details
Test Architect	2021-02-06 2021-02-07 (Sat-Sun) Weekend batch	View Details
Test Architect	2021-02-13 2021-02-14 (Sat-Sun) Weekend batch	View Details

Leave a Reply

Your email address will not be published. Required fields are marked *

Selenium IDE

Updated on January 7, 2021 |
This part of the Selenium tutorial will teach you about the Selenium IDE, the installation process for Selenium IDE, so you know exactly how to go about installing Selenium IDE.

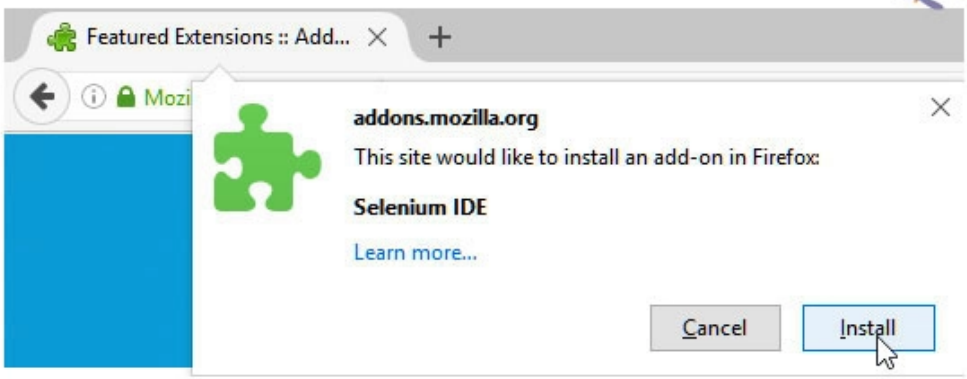
How to Download & Install Selenium IDE for Firefox

In this chapter, you will get to know more about Selenium IDE. Read below:
Selenium IDE (Integrated Development Environment) is a plug-in for the Mozilla Firefox web browser for running Selenium scripts. It allows software testing professionals to control the tests, such as record, edit, play, debug, etc. It consists of the core of Selenium which executes the tests. It does not just control the tests, but it provides an environment for the scripts to run.

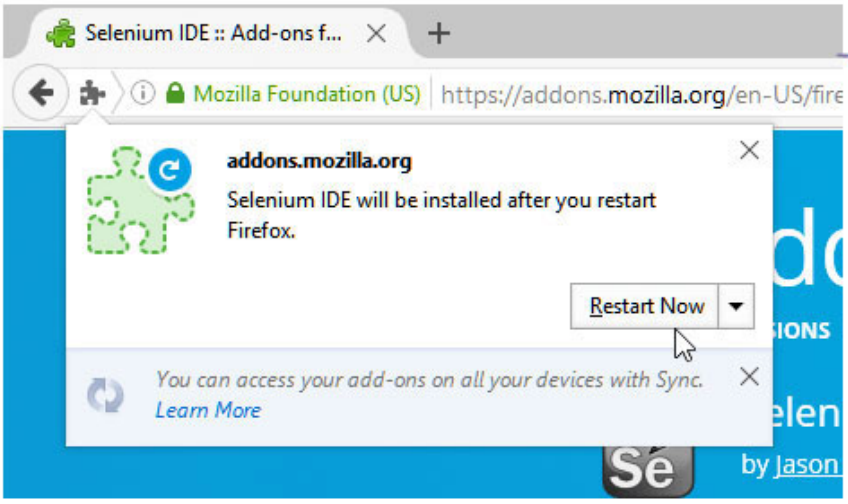
Watch this Selenium Tutorial for beginner

Selenium IDE installation procedure

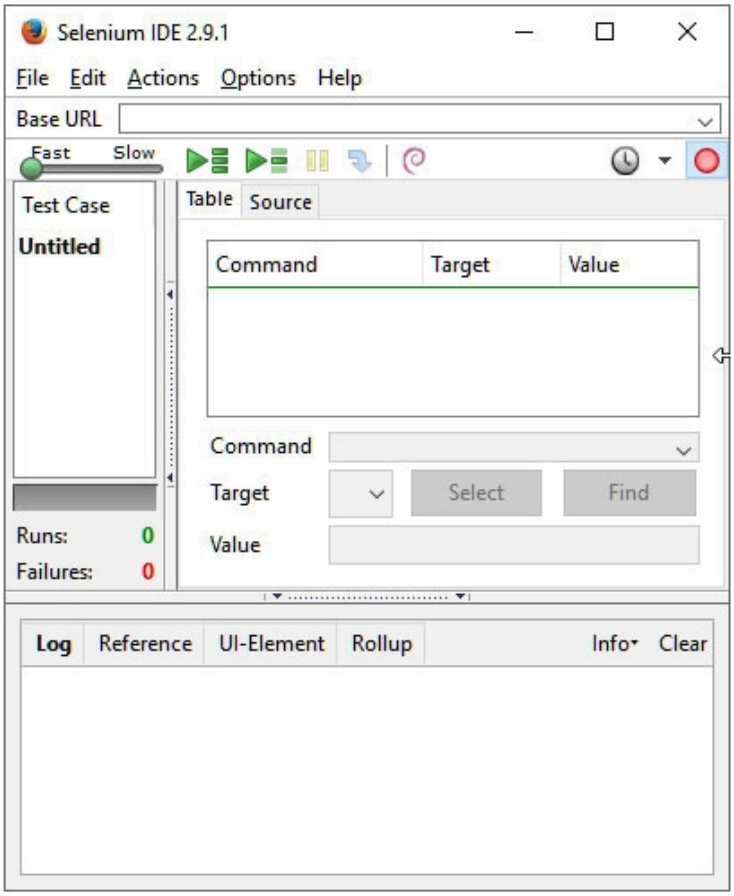
In order to download and install the Selenium IDE follow given steps:
Step 1: Launch Firefox browser and visit the following link https://addons.mozilla.org/en-US/firefox/addon/selenium-ide/ and click on 'Add to Firefox' button.
Step 2: Click on the Install button and let it finish the procedure.



Step 3: A pop-up window will appear. Click on 'Restart Now' thereafter.



Step 4: Launch Selenium IDE by going to the Firefox Menu button, click on developer and then Selenium IDE.



Learn Selenium with industry experts from Intellipaat's online Selenium training program.

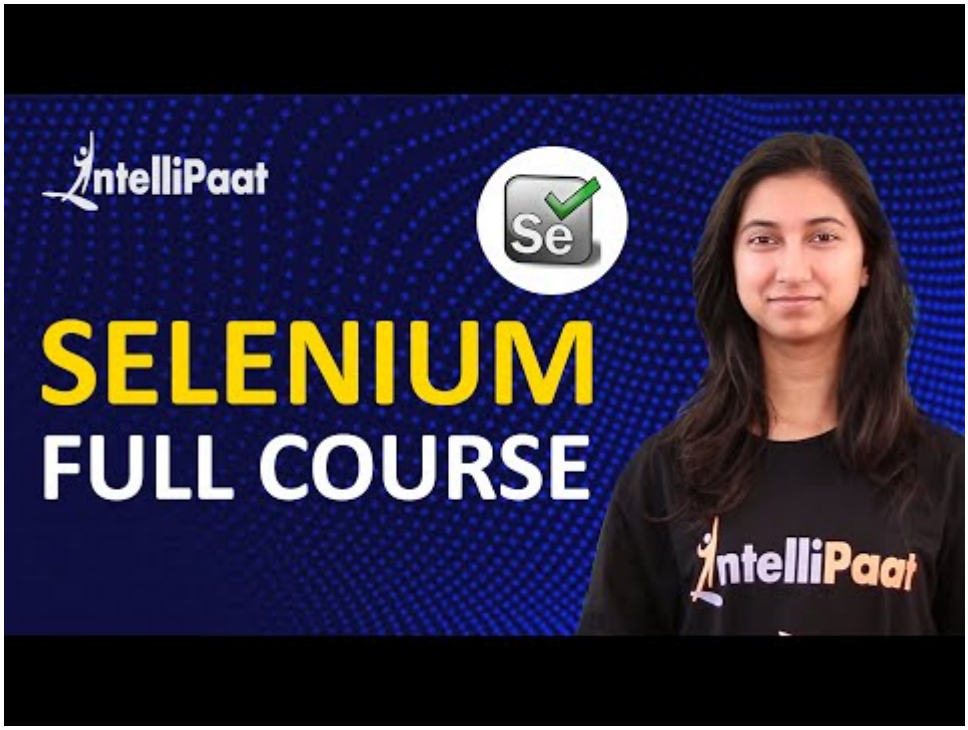
Previous Next

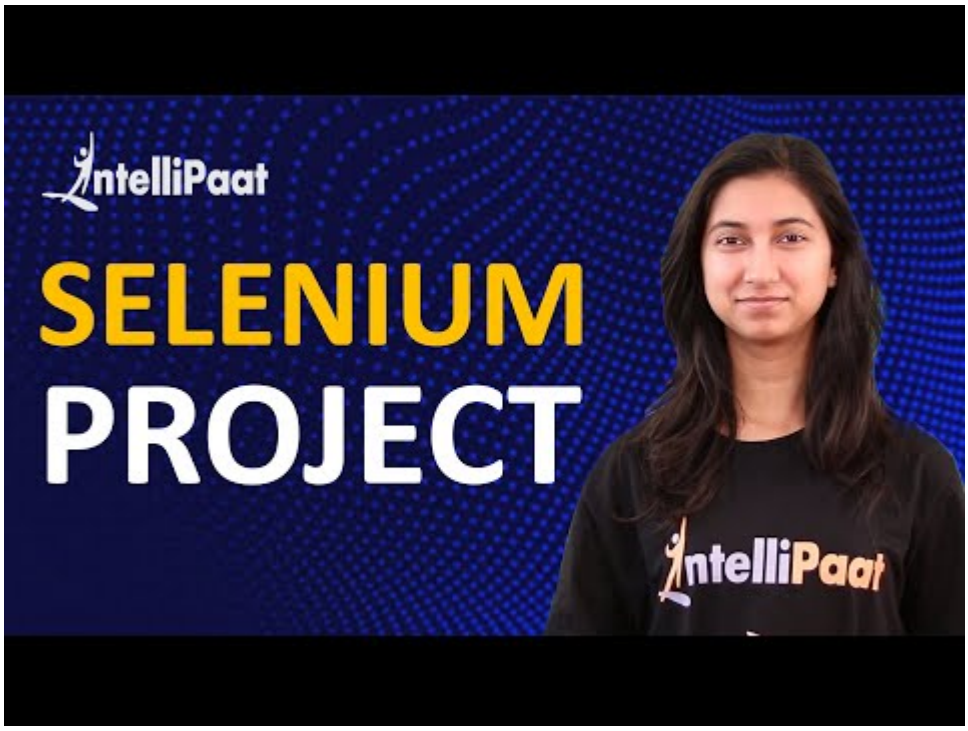
Recommended Videos


IntelliPaat
SELENIUM
TUTORIAL
MASTER TOP AUTOMATION
TESTING TOOL
IntelliPaat


IntelliPaat
SELENIUM
TUTORIAL
FOR BEGINNERS
IntelliPaat


IntelliPaat
Selenium automation
Testing
Redefining Learning
Intellipaat
Subscribe
Se


IntelliPaat
Se
SELENIUM
FULL COURSE
IntelliPaat



Course Schedule

Name	Date	
Test Architect	2021-01-30 2021-01-31 (Sat-Sun) Weekend batch	View Details
Test Architect	2021-02-06 2021-02-07 (Sat-Sun) Weekend batch	View Details
Test Architect	2021-02-13 2021-02-14 (Sat-Sun) Weekend batch	View Details

Leave a Reply

Your email address will not be published. Required fields are marked *

Recording of a Test

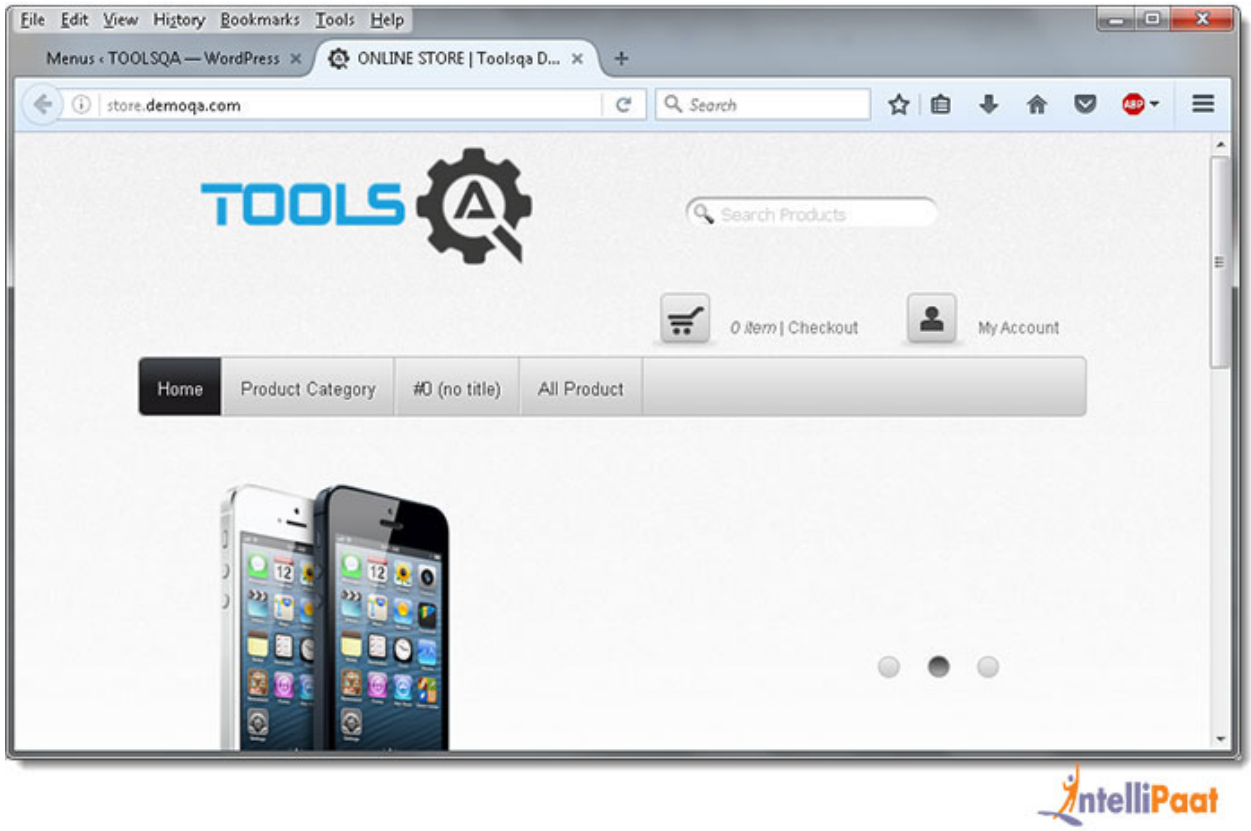
Updated on January 8, 2021 |
This section of the Selenium tutorial includes recording of a test case using Selenium, you will start the process for recording a Selenium test and successfully will be able to record the test.

Steps to record Selenium test

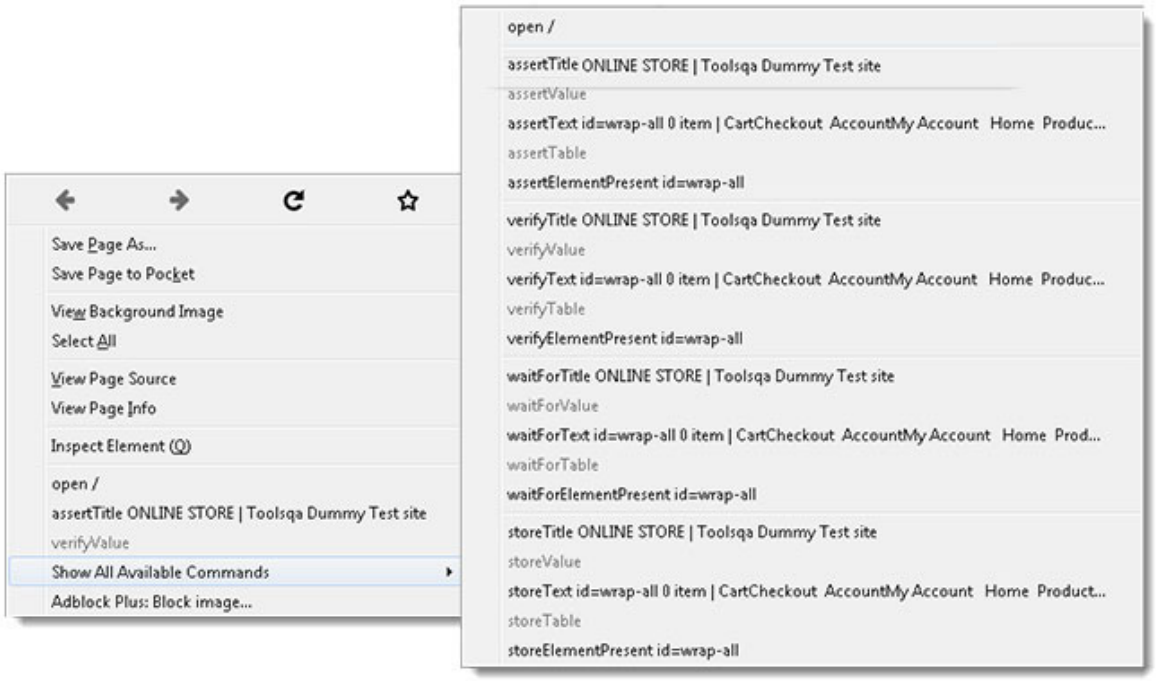
In order to record a test, follow the given steps:
Step 1 : Open Mozilla Firefox web browser, go to tools and click on Selenium IDE. There type the link
http://store.demoqa.com

Watch How to Write & Run a Test Case in Selenium Tutorial:

Now start the recoding procedure and initiate the test case.
Step 2 : After entering the link, you will land on a page

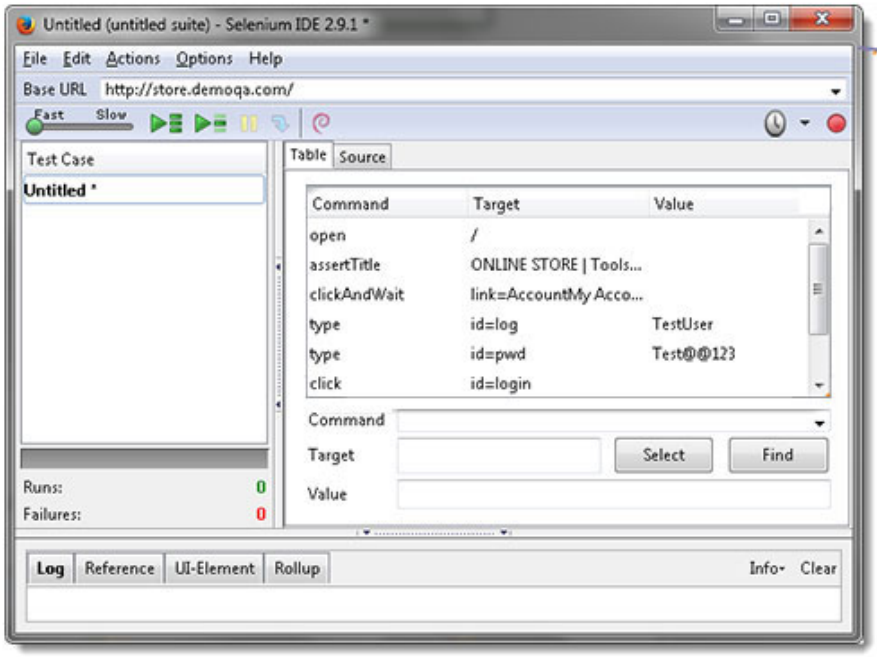


Step 3 : Now right click on the blank space and select 'Show Available Commands' and click on '**assertTitle ONLINE STORE | Toolsqa Dummy Test site**'.
Want to know how to test web applications using Selenium? Read this informative blog!



Step 4 : Enter credentials when asked. enter Username and password and hit the 'Login' button.

Step 5 : You can stop the recording by clicking the 'Record' button.

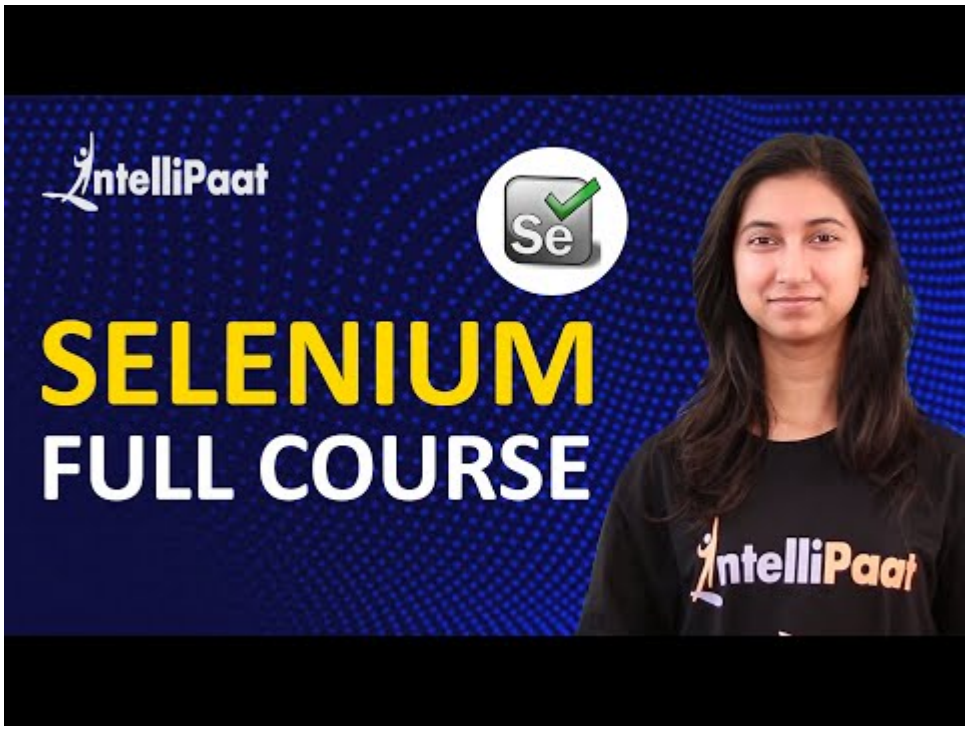


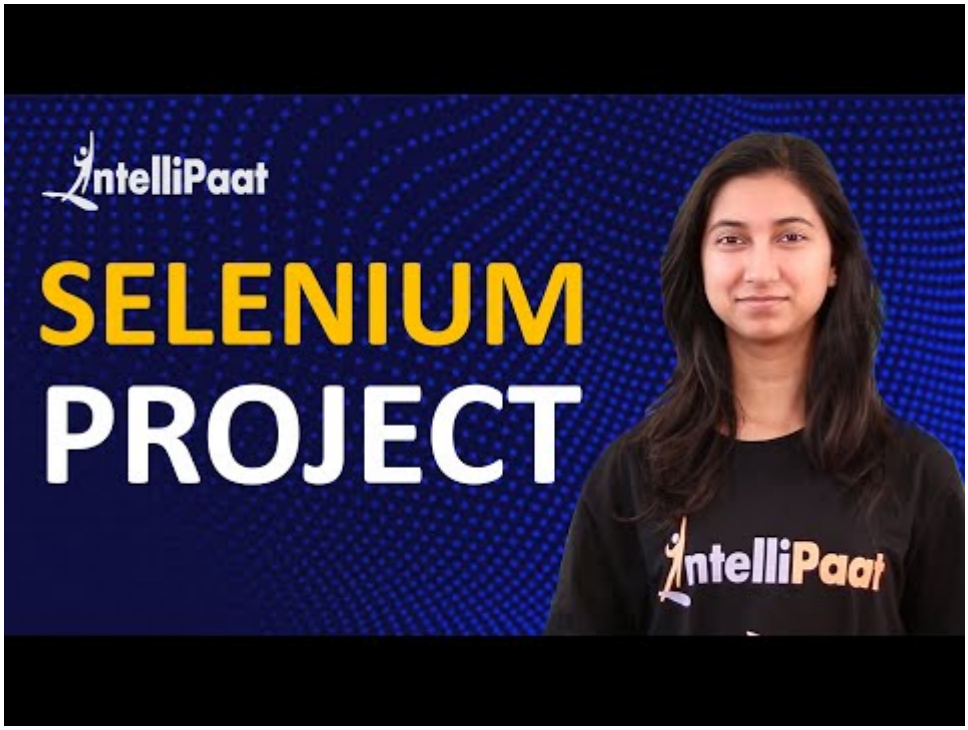
Previous Next

Recommended Videos


IntelliPaat
SELENIUM
TUTORIAL
FOR BEGINNERS
IntelliPaat


IntelliPaat
Selenium automation
Testing
Redefining Learning
Intellipaat
Subscribe





Course Schedule

Name	Date	
Test Architect	2021-01-30 2021-01-31 (Sat-Sun) Weekend batch	View Details
Test Architect	2021-02-06 2021-02-07 (Sat-Sun) Weekend batch	View Details

Test Architect	2021-02-13 2021-02-14 (Sat-Sun) Weekend batch	View Details

Leave a Reply

Your email address will not be published. Required fields are marked *

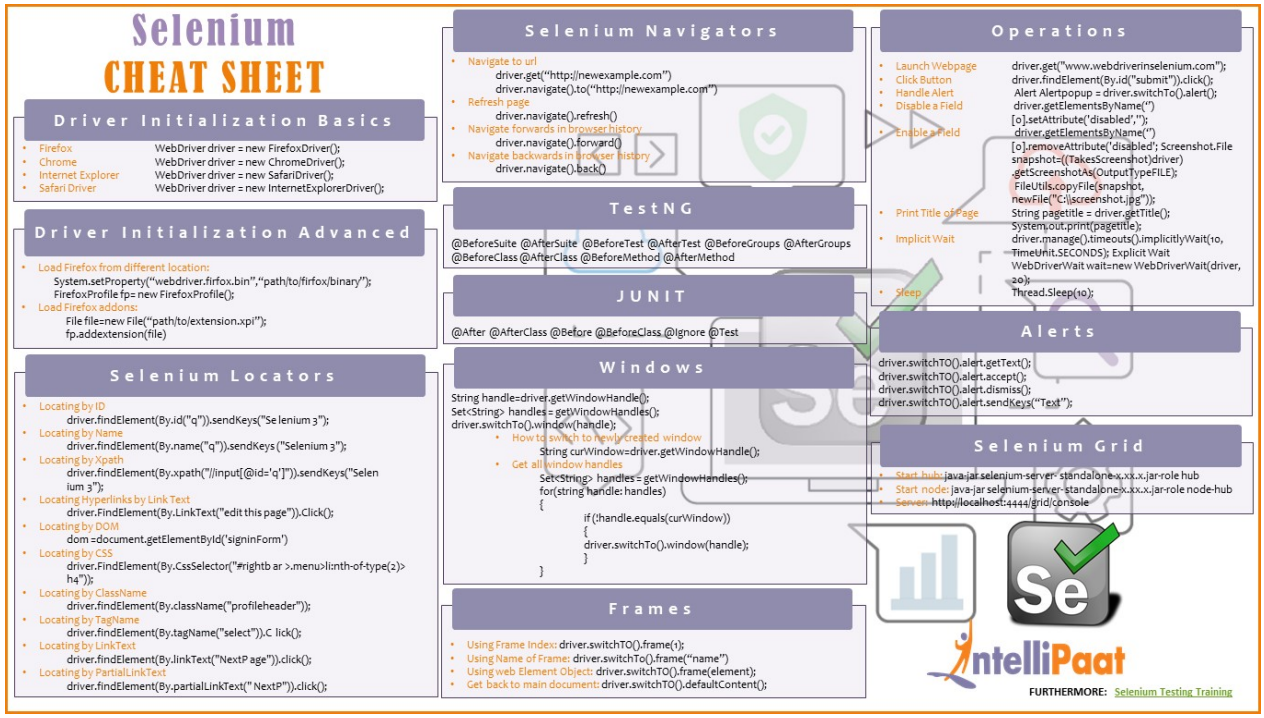
Selenium Cheat Sheet

intellipaat.com/blog/tutorial/selenium-tutorial/selenium-cheat-sheet/

Selenium User Handbook

Are you looking for a quick reference to turn to every time you need to look up the basic operations of selenium? Well if you are then this is just what you need.
We at Intellipaat understand that it could be a bit of a hassle to remember every aspect of technology and that is why we have come up with a Selenium cheat sheet to assist our learners whenever they need a handy reference containing all the basics of selenium.

We assume that you have a basic understanding of selenium if you are referring to this cheat sheet.
Download the printable PDF of Selenium cheat sheet



What is selenium?

Selenium is an open-source framework to automate and perform software testing such as smoke tests, integration tests, etc. on web applications. It provides a playback/recording tool and domain-specific language.

Get started with Selenium:

Driver Initialization Basics:

- Firefox WebDriver driver = new FirefoxDriver();
- Chrome WebDriver driver = new ChromeDriver();
- Internet Explorer WebDriver driver = new SafariDriver();
- Safari Driver WebDriver driver = new InternetExplorerDriver();

Driver Initialization Advanced:

A. Load firefox from different location:

```
System.setProperty(“webdriver.firfox.bin”,“path/to/firfox/binary”);
FirefoxProfilefp= new FirefoxProfile();
```

B. Load firfox addons

```
File file=new File(“path/to/extension.xpi”);
fp.addextension(file)
```

Selenium Locators:

Selenium locators are used to find and match the webpage elements that selenium interacts with. Following are some locators in selenium:

Locating by ID

```
driver.findElement(By.id("q")).sendKeys("Se lenium 3");
```

Locating by Name

```
driver.findElement(By.name("q")).sendKeys ("Selenium 3");
```

Locating by Xpath

```
driver.findElement(By.xpath("//input[@id='q'])).sendKeys("Selenium 3");
```

Locating Hyperlinksby Link Text

```
driver.FindElement(By.LinkText("edit this page")).Click();
```

Locating by DOM

```
dom =document.getElementById('signinForm')
```

Locating by CSS

```
driver.FindElement(By.CssSelector("#rightbar> .menu >li:nth-of-type(2) > h4"));
```

Locating by ClassName

```
driver.findElement(By.className("profileheader"));
```

Locating by TagName

```
driver.findElement(By.tagName("select")).C lick();
```

Locating by LinkText

```
driver.findElement(By.linkText("NextP age")).click();
```

Locating by PartialLinkText

```
driver.findElement(By.partialLinkText(" NextP")).click();
```


Selenium Navigators:

The navigator interface in selenium helps in moving backwards and forwards in the browser's history. Following are some navigator commands you can use:

Navigate to url

```
driver.get("http://newexample.com")
driver.navigate().to("http://newexample.com")
```

Refresh page

```
driver.navigate().refresh()
```

Navigate forwards in browser history

```
driver.navigate().forward()
```

Navigate backwards in browser history

```
driver.navigate().back()
```

TestNG:

TestNG is an open source framework for automated testing. The NG in TestNG stands for Next Generation. It is similar to Junit but with more functionality to offer. Following are the TestNG annotations:

- **@test:** This annotation marks a class or method as a part of a test
- **@BeforeSuite:** This annotation makes sure that the method only run once before all the tests have run
- **@AfterSuite:** This annotation makes sure that the method runs once after the execution of all the tests
- **@BeforeTest:** This annotation will make sure that the method marked with this annotation runs before the first method annotated with @test

- **@AfterTest:** This annotation will make sure that the method marked with this annotation runs after all the methods annotated with @test execute all the classes inside <test> tag in the testng.xml file.
- **@BeforeGroups :** A method annotated with this annotation will run before all the first test methods run in that specific group
- **@AfterGroups:** A method annotated with this annotation will run after all the test methods run in that specific group
- **@BeforeClass:** A method annotated with this annotation will run only once per class and before all the first test methods run
- **@AfterClass:** A method annotated with this annotation will run only once per class and after all the test methods run
- **@BeforeMethod:** A method annotated with this annotation will run before every @test annotated method
- **@AfterMethod:** A method annotated with this annotation will run after every @test annotated method

JUNIT

JUNIT is a framework used to perform unit level testing. Following are the JUNIT annotations:

- **@Test:** test method to run with public void return type
- **@After:** method to run after test method
- **@AfterClass:** method to run before test method
- **@Before:** method to run before test method
- **@BeforeClass:** method to run once before any of the test methods in the class have been executed
- **@Ignore:** This annotation is used to ignore a method

Windows:

sometimes the web applications may have multiple frames or windows. Selenium assigns each window a unique alphanumeric id which is called window handle. Selenuim then uses the id to switch control among windows.

```
String handle=driver.getWindowHandle();
Set<String> handles = getWindowHandles();
driver.switchTo().window(handle);
```

How to switch to newly created window

```
 String curWindow=driver.getWindowHandle();
```

Get all window handles

```
Set<String> handles = getWindowHandles();
For(string handle: handles){
If (!handle.equals(curWindow))
{driver.switchTo().window(handle);
}
}
```

Frames

Using Frame Index:

```
driver.switchTO().frame(1);
```

Using Name of Frame:

```
driver.switchTo().frame(“name”)
```

Using web Element Object:

```
driver.switchTO().frame(element);
```

Get back to main document:

```
driver.switchTO().defaultContent();
```

Operations:

In selenium there are certain operations that can be performed on the web elements. Following is the list of those operations along with their respective syntax:

Launch Webpage:

```
get("www.webdriverinselenium.com");
```

Click Button:

```
findElement(By.id("submit")).click();
```

Handle Alert:

```
AlertAlertpopup = driver.switchTo().alert();
```

Disable a Field:

```
getElementsByName('') [0].setAttribute('disabled', '')
```

Enable a Field :

```
getElementsByName('') [0].removeAttribute('disabled');
```

Screenshot :

```
File snapshot = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(snapshot, new File("C:\\screenshot.jpg"));
```

Print the Title of the Page:

```
String pagetitle = driver.getTitle();
System.out.print(pagetitle);
```

Implicit Wait:

```
manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```

Explicit Wait:

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

Sleep :

```
Thread.Sleep(10);
```

Watch this Selenium Projects For Beginners Tutorial

Alerts:

Sometimes a message box pops up on the screen to display some kind of notification to the user or maybe an ask for a permission or to display a warning etc. These messages are called alerts. Alert interface provides some ways to handle the alerts in selenium:

Capture the alert message:

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

Click on the ‘OK’ button of the alert:

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

Click on the ‘Cancel’ button of the alert:

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

Send some data to alert box:

```
driver.switchTO().alert.sendKeys(“Text”);
```

Selenium Grid:

Selenium Grid helps selenium run multiple tests across different operating systems, browsers and machines in parallel.

Start hub:

```
java-jar selenium-server- standalone-x.xx.x.jar-role hub
```

Start node:

```
java-jar selenium-server- standalone-x.xx.x.jar-role node-hub
http://localhost:4444/grid/register
```

Server:

```
http://localhost:4444/grid/console
```

Download a Printable PDF of this Cheat Sheet

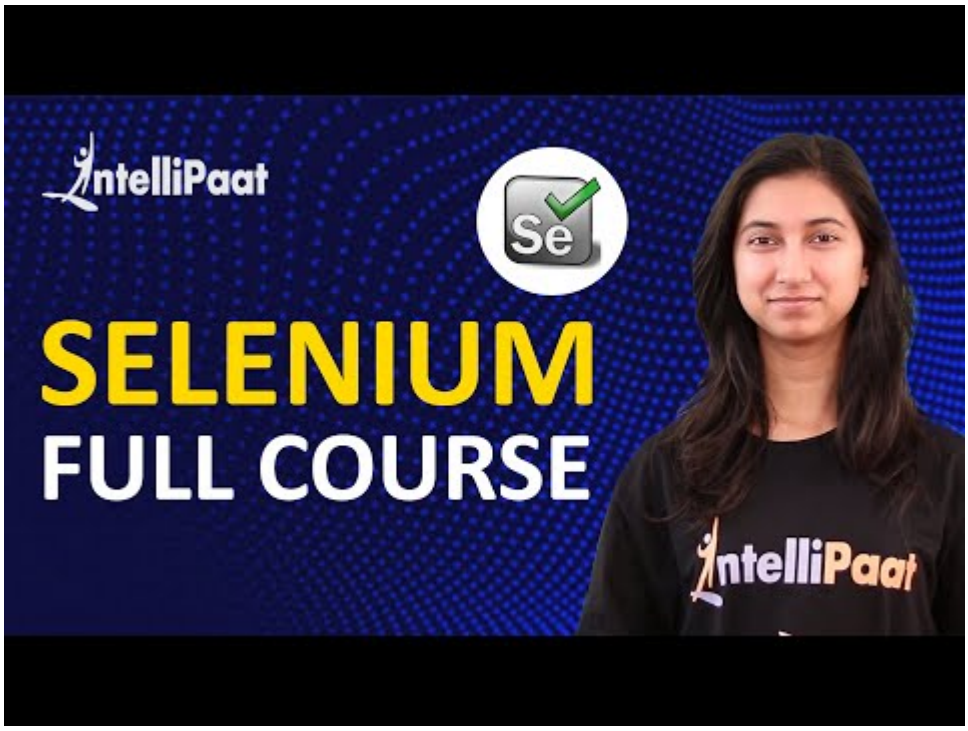
This would be all for the Selenium cheat sheet. In case you are looking to learn Selenium in-depth then you should definitely check out the Selenium training provided by Intellipaat. In this online training, you will get to learn the automation testing framework for web applications, TDD, selenium architecture, JaCoCo, TestNG, Sikuli. You will work on real-life projects and assignments and prepare yourself for Certified Selenium Professional certification in this Selenium training. On top of that, you will have 24*7 technical support from our experts here at Intellipaat.

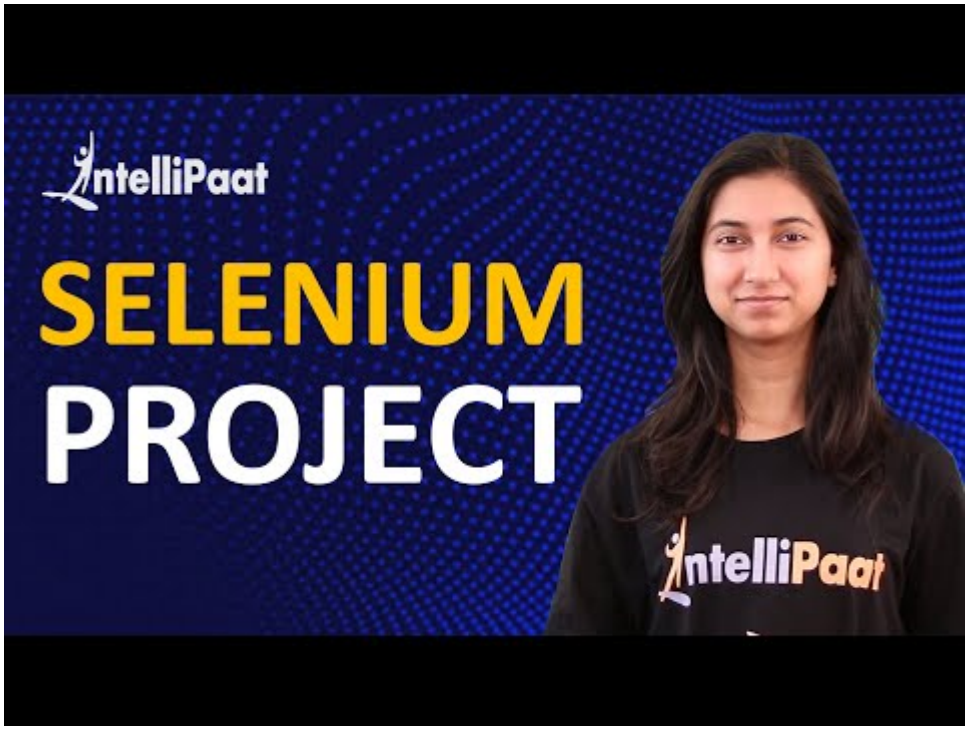
Previous Next

Recommended Videos


IntelliPaat
SELENIUM
TUTORIAL
FOR BEGINNERS
IntelliPaat


IntelliPaat
Selenium automation
Testing
Redefining Learning
Intellipaat
Subscribe





Course Schedule

Name	Date	
Test Architect	2021-01-23 2021-01-24 (Sat-Sun) Weekend batch	View Details
Test Architect	2021-01-30 2021-01-31 (Sat-Sun) Weekend batch	View Details

Test Architect	2021-02-06 2021-02-07 (Sat-Sun) Weekend batch	View Details

Leave a Reply

Your email address will not be published. Required fields are marked *

Features of Selenium

Let us know what is so special about Selenium:

Combination of tool and DSL – Selenium is an absolute combination of tools and DSL (Domain Specific Language) in order to carry out various types of tests. It allows you to record the tests carried out through the browser. It supports multiple web browsers like Internet Explorer, Safari, Firefox, Chrome, etc.

Uses a rich language for tests – Selenium uses DSL in order to test the web applications. This language includes 200 commands and is an easy programming language to learn.

Watch this Selenium Tutorial video

A flexible language – Once the test cases are prepared, they can be executed on any operating system like Linux, Macintosh, etc.

Reduce test execution time – Selenium supports parallel test execution that reduce the time taken in executing parallel tests.

Lesser resources required – Selenium requires lesser resources when compared to its competitors like UFT, RFT, etc.

Drawbacks of Selenium

Incomplete solution – Selenium requires third party frameworks in order to completely automate the testing of web applications.

Requires high skills – Though it supports multiple programming languages, it requires a high-level proficiency to deal with it effectively.

Hard to modify codes – The scripts written in **Selenese** is not user-friendly which makes it hard to modify the codes.

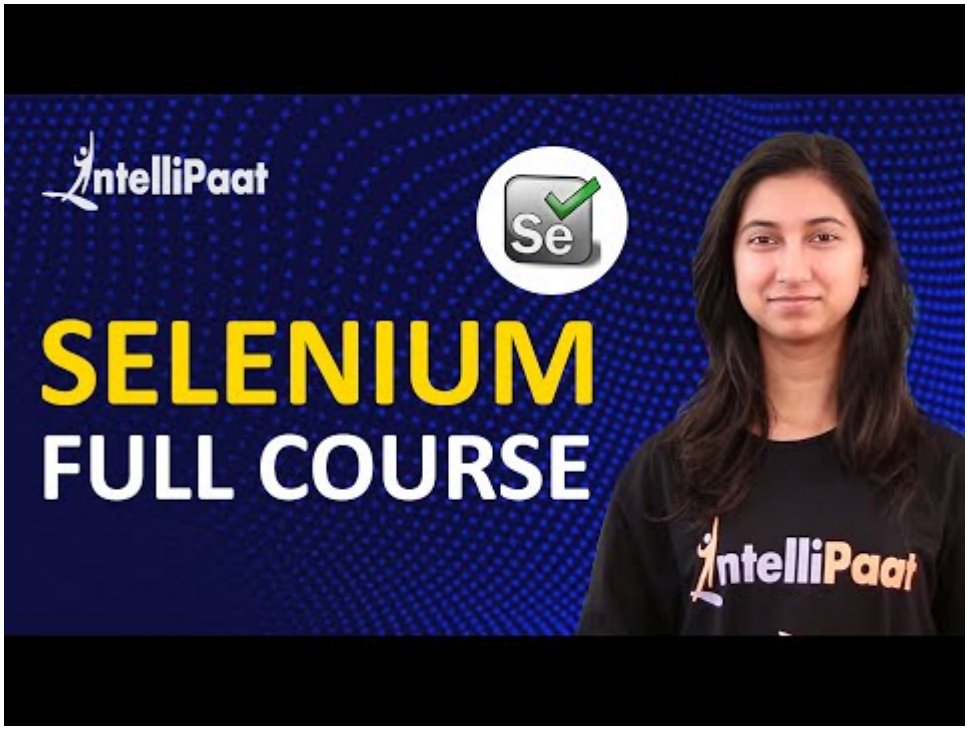
Tougher to support other browsers – Selenium faces difficulties when tried to implement in any browser other than Firefox.

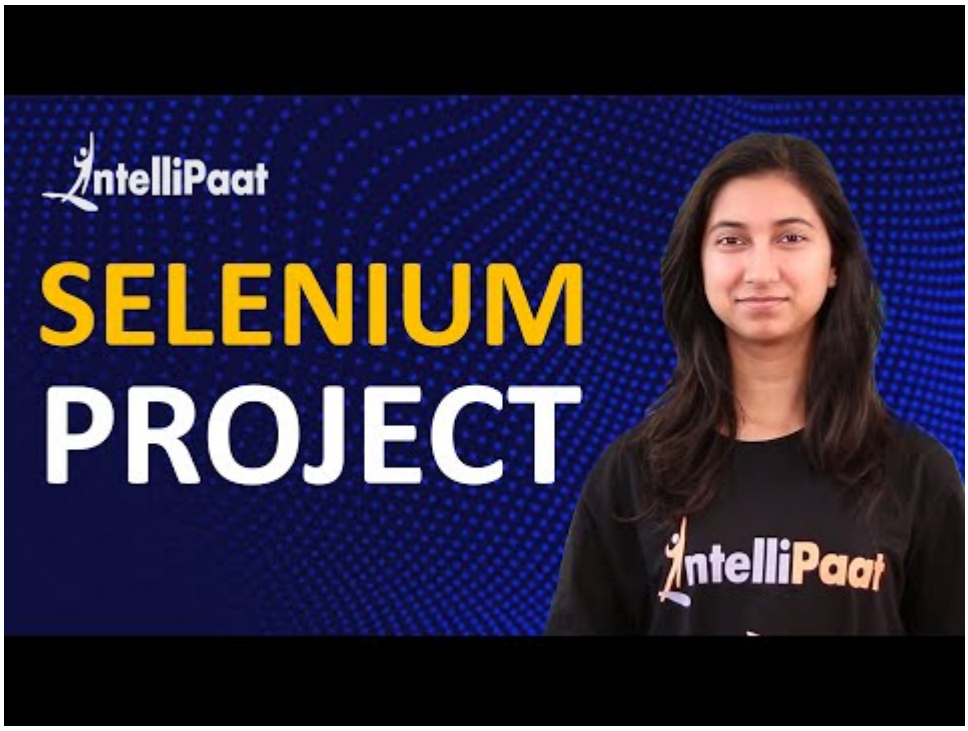
Enrol with online Selenium training course and grab a high-paying software testing job today!

Previous Next

Recommended Videos


IntelliPaat
Selenium automation
Testing
IntelliPaat
Redefining Learning
Intellipaat
Subscribe
Se


IntelliPaat
Se
SELENIUM
FULL COURSE
IntelliPaat



Course Schedule

Name	Date	
Test Architect	2021-01-30 2021-01-31 (Sat-Sun) Weekend batch	View Details
Test Architect	2021-02-06 2021-02-07 (Sat-Sun) Weekend batch	View Details
Test Architect	2021-02-13 2021-02-14 (Sat-Sun) Weekend batch	View Details

Leave a Reply

Your email address will not be published. Required fields are marked *

TestNG in Selenium

intellipaat.com/blog/tutorial/selenium-tutorial/testng-in-selenium/

TestNG in Selenium is a Java testing framework, inspired by JUnit and NUnit. It overcomes the constraints and disadvantages of JUnit and introduces an entirely new set of properties, making TestNG more powerful and easy to use. The suffix 'NG' stands for Next Generation, signifying the new functionalities that TestNG brings to the table. From simple unit testing to complex integrated testing, it is designed to simplify all your testing requirements such as functional testing, regression, end-to-end testing, and more.

As part of this TestNG in Selenium tutorial, you will learn the following topics:

Check out this Intellipaat video to know 'What is TestNG in Selenium?':

Before moving ahead and knowing what exactly TestNG in Selenium is all about, let us understand why did TestNG come into existence?

Why did TestNG come into existence?

You must be wondering when you already have Selenium framework, why do you need TestNG with Selenium WebDriver for automation testing? Well, the answer is, since Selenium does not have any built-in hardware or framework for generating test reports, you require the help of an external framework like TestNG to fulfil the need for generating test reports and simplifying testing requirements such as functional testing, regression, end-to-end testing, and more.

TestNG in Selenium: An Overview

In Selenium, there are mainly two testing frameworks available and they are:

1) JUnit

2) TestNG

What is TestNG in Selenium?

TestNG is an open-source testing framework where NG stands for 'Next Generation.' It is architected to simplify a broad range of testing needs starting from unit testing to integrated system testing. Initially, both JUnit and TestNG were designed solely for unit testing. TestNG is inspired by JUnit Java platform and NUnit .NET platform, and some new functionalities were introduced in TestNG, making it more powerful and easy to use than the JUnit testing framework.

If you want to use Selenium with .NET, then you have to use the NUnit testing framework, since NUnit supports the .NET platform.

In this section of the TestNG in Selenium tutorial, let us now know about the advantages of TestNG which makes it more powerful over JUnit.

Advantages of TestNG over JUnit

1. TestNG Annotations are used to create test cases easily.
2. Test cases can be 'grouped,' 'prioritized,' and 'executed' more efficiently.
3. It supports parameterization.
4. It supports data-driven testing using Data Providers.
5. It can generate HTML test reports of the results representing: the number of test cases runs, the number of test cases failed, the number of test cases skipped
6. It effortlessly supports integration with various other tools and plugins like Eclipse IDE and built automation tools like Ant and Maven.
7. It supports parallel execution.
8. Logs can be generated.
9. In TestNG, there is no need to state @AfterClass and @BeforeClass in a project, which is present in JUnit.
10. You can specify any test method name in TestNG as method's name constraint is not present in TestNG like it is in JUnit.
11. TestNG supports the following three additional setup and teardown levels: @Before/AfterSuite, @Before/AfterTest, and @Before/AfterGroup. TestNG goes beyond the idea of just writing @Test annotated methods and allows you to define these methods that will be run after or before your test suites, test groups, or test methods. This is very useful for your Selenium tests because you can create a Selenium server and browser instance before you start running your test suite.
12. TestNG in Selenium does not require to extend any class; hence, no need for the inheritance functionality.
13. TestNG allows us to define the dependent test cases.
14. TestNG in Selenium allows us to execute test cases based on the group. Let's take a scenario where you have created two sets of groups 'Regression' and 'Sanity.' If you want to execute the test cases under Sanity group, then you can do so easily with the TestNG framework.

In this TestNG in Selenium tutorial, let us now see how to install TestNG.

Installation of TestNG

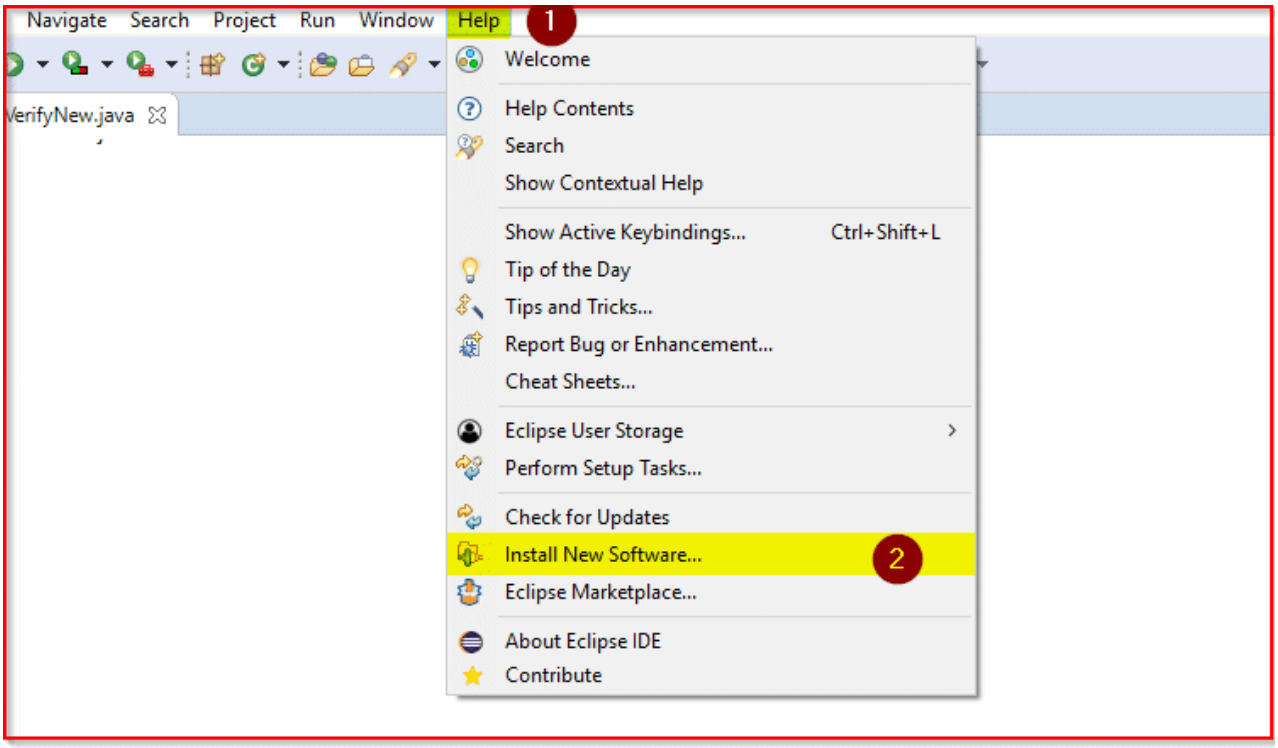
Since Selenium does not have any built-in hardware for generating test reports, you need the help of an external framework like TestNG to fulfill the need for generating test reports and simplifying all your testing requirements such as functional testing, regression, end-to-end testing, and more.

Now that you know the need for installing TestNG, let us get started with installing this testing framework.

For this, it is assumed that you have **Java JDK** and **Eclipse IDE for Java Developers** already installed in your system.

Step 1: In the **Eclipse IDE for Java Developers**

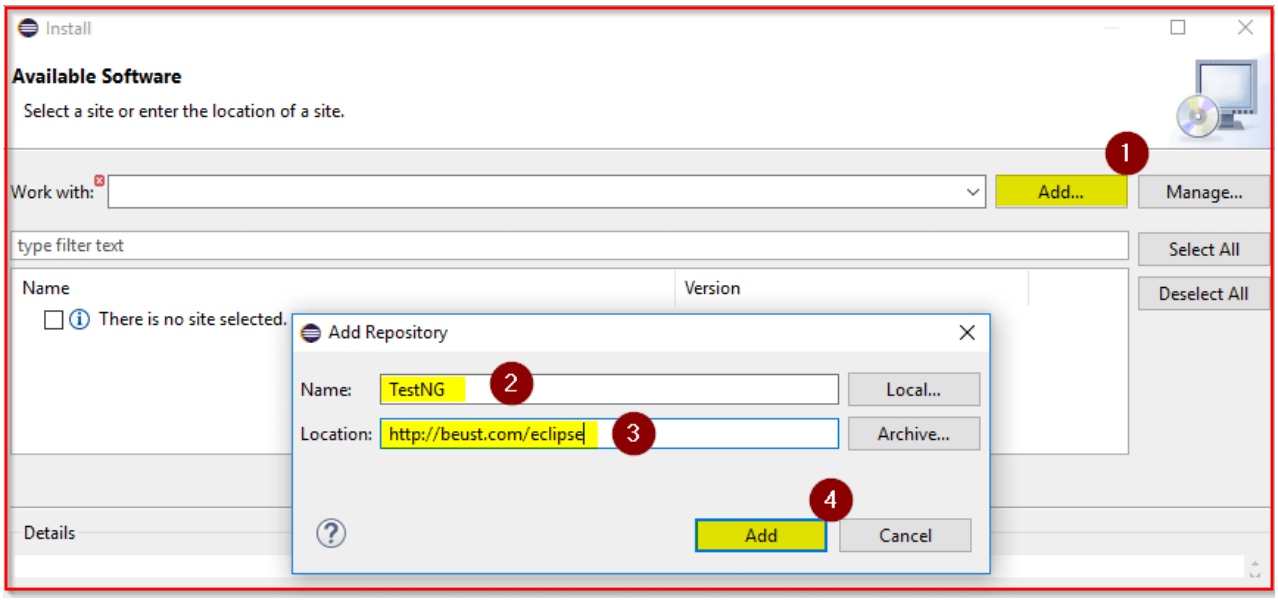
1. Click on the **Help** menu
2. Click on **Install New Software**



Step 2: The below-shown window will popup

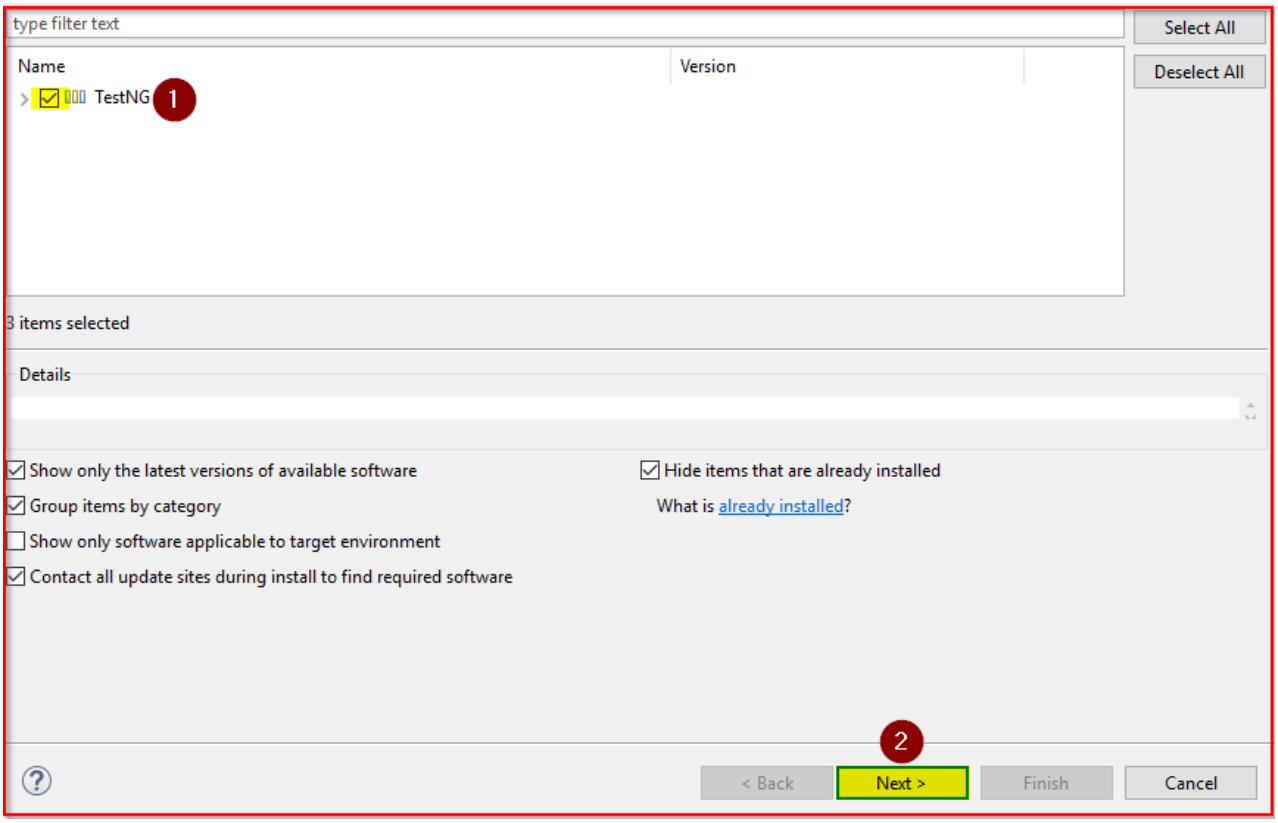
1. Click on the **Add** button

2. In the **Name** text box, type '**TestNG**'
3. In the **Location** text box, type the URL '**http://beust.com/eclipse**'
4. Click on **Add**

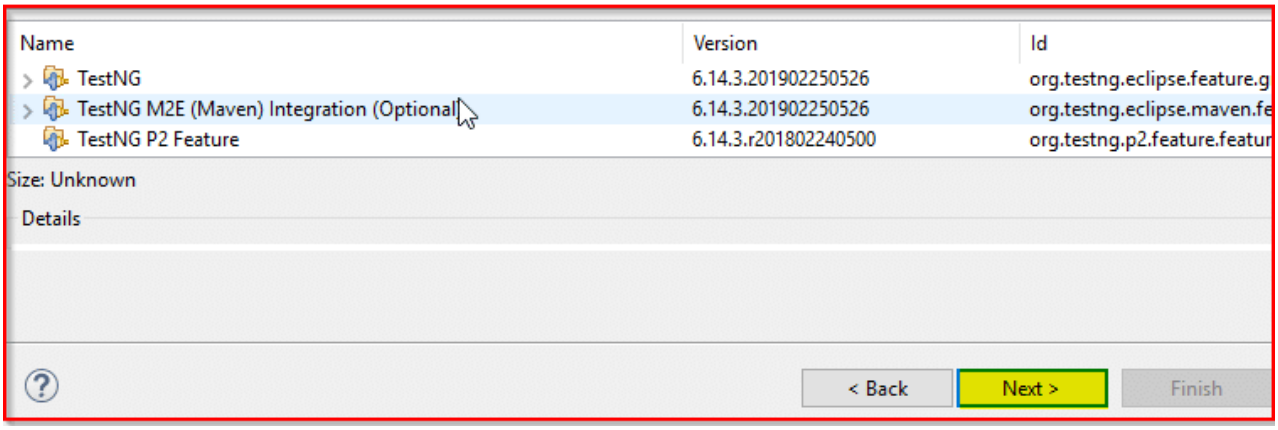


Step 3: The below-shown window will popup:

1. Select **TestNG**
2. Click on **Next and then on Finish**

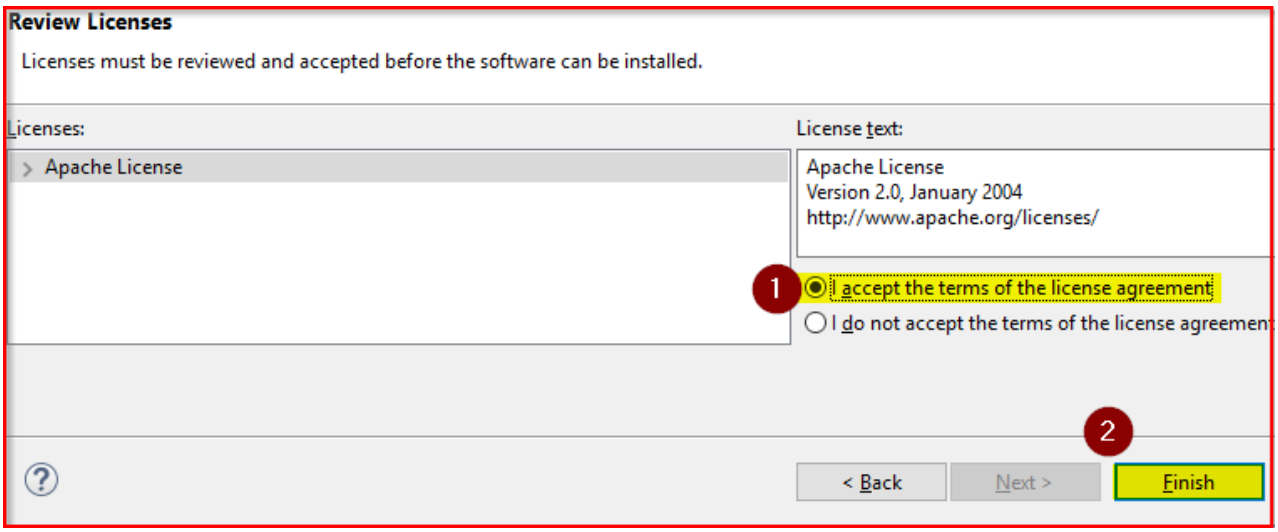


Step 4: When the below-shown window pops up, click on **Next**



Step 5: Then, a window as shown below will popup:

1. Click on the '**I accept the terms of the license agreement**' radio button
2. Click on **Finish**



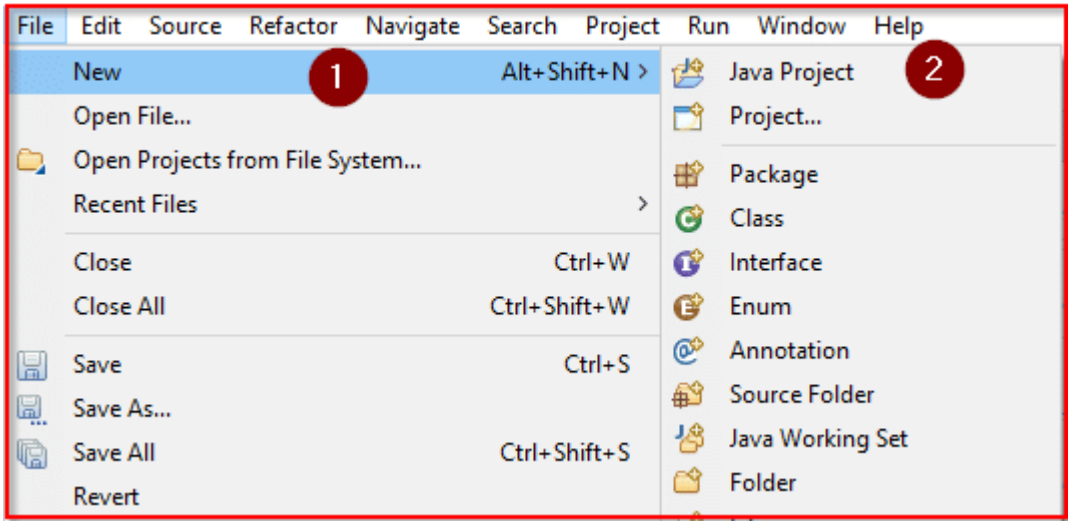
Now, you have to **restart** the Eclipse IDE, select **'Yes'** on the prompt box asking you to restart the Eclipse IDE.

There you go. You have successfully installed TestNG on the Eclipse IDE!

In this section of the TestNG in Selenium tutorial, let us move ahead and see how you can write your first multiple TestNG test cases in a single configuration file (**testng**.xml) in Eclipse:

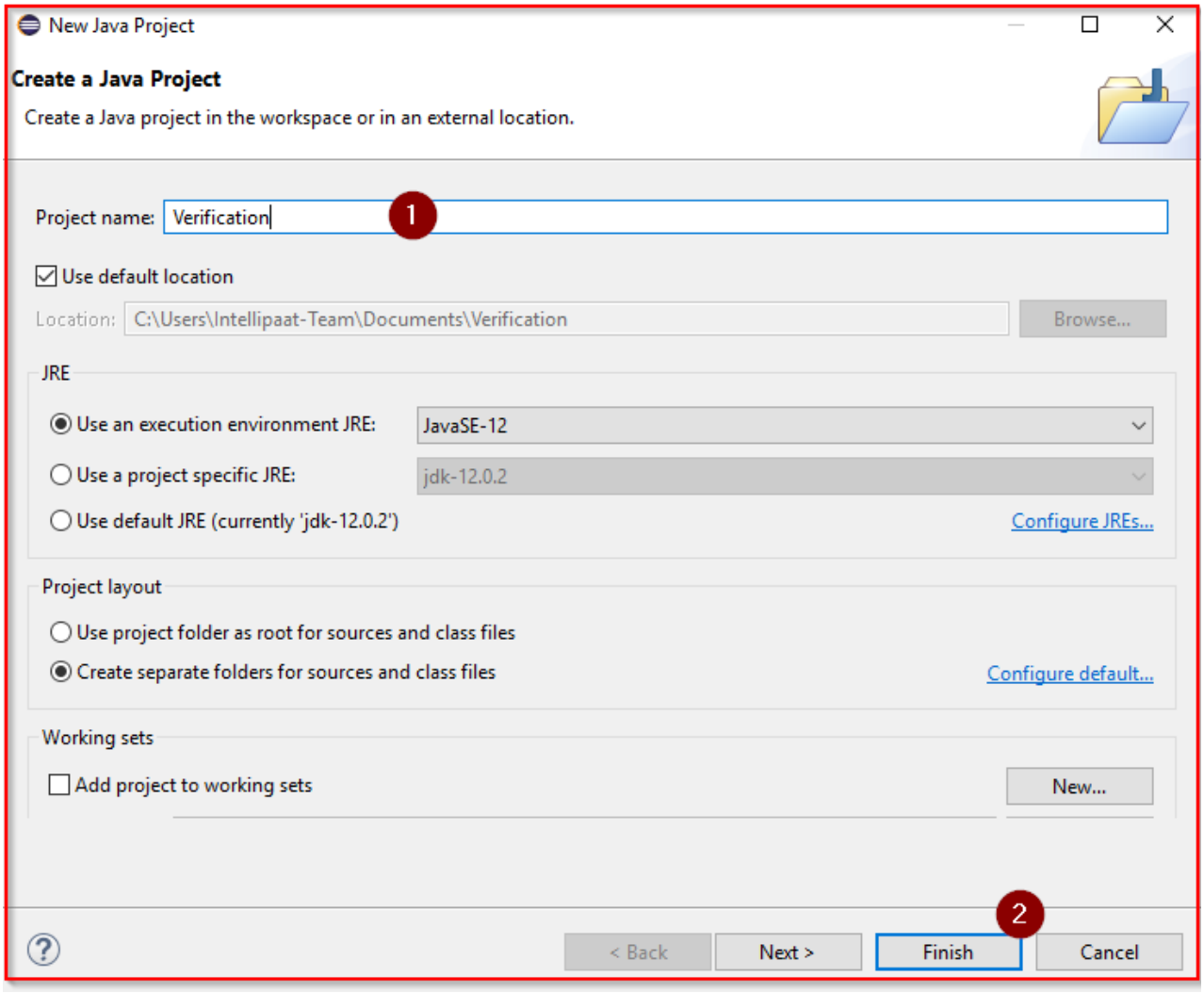
Step 1: In the Eclipse IDE, click on the **File** menu

1. Click on **New**
2. Click on **Java Project**

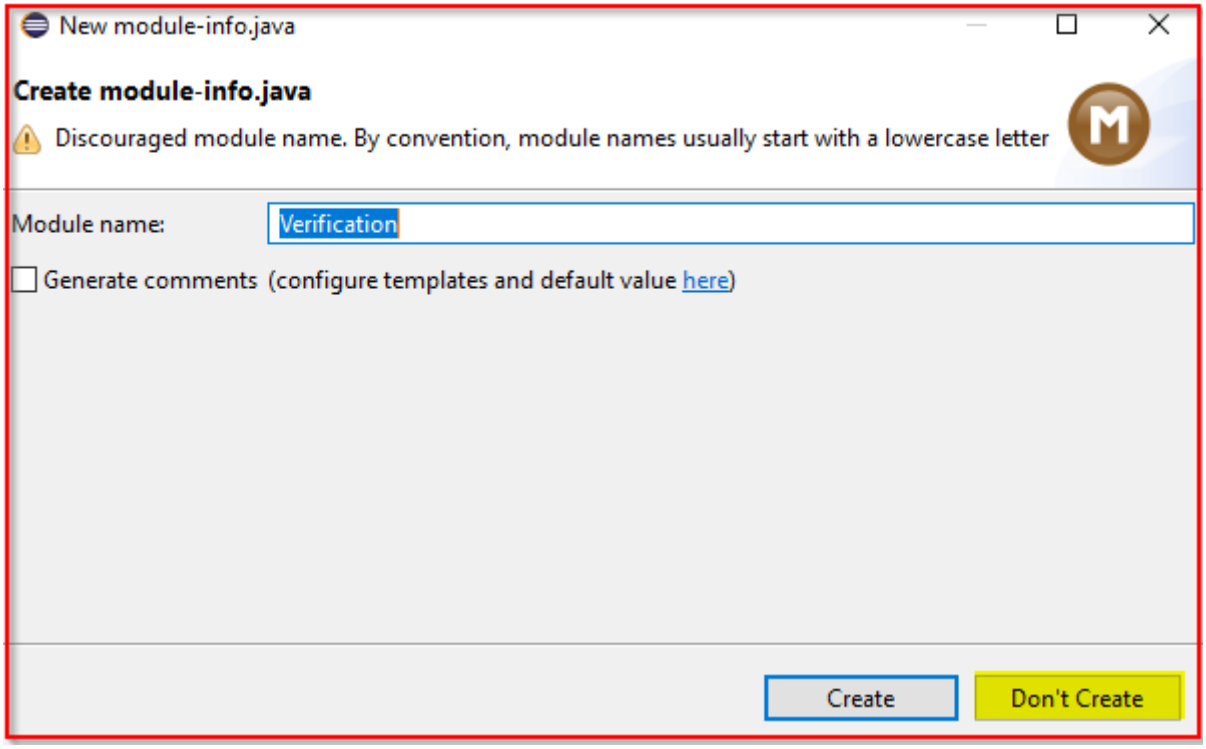


Step 2:

1. Give your project name as '**Verification**.' You can give any name as per your wish
2. Click on **Finish**

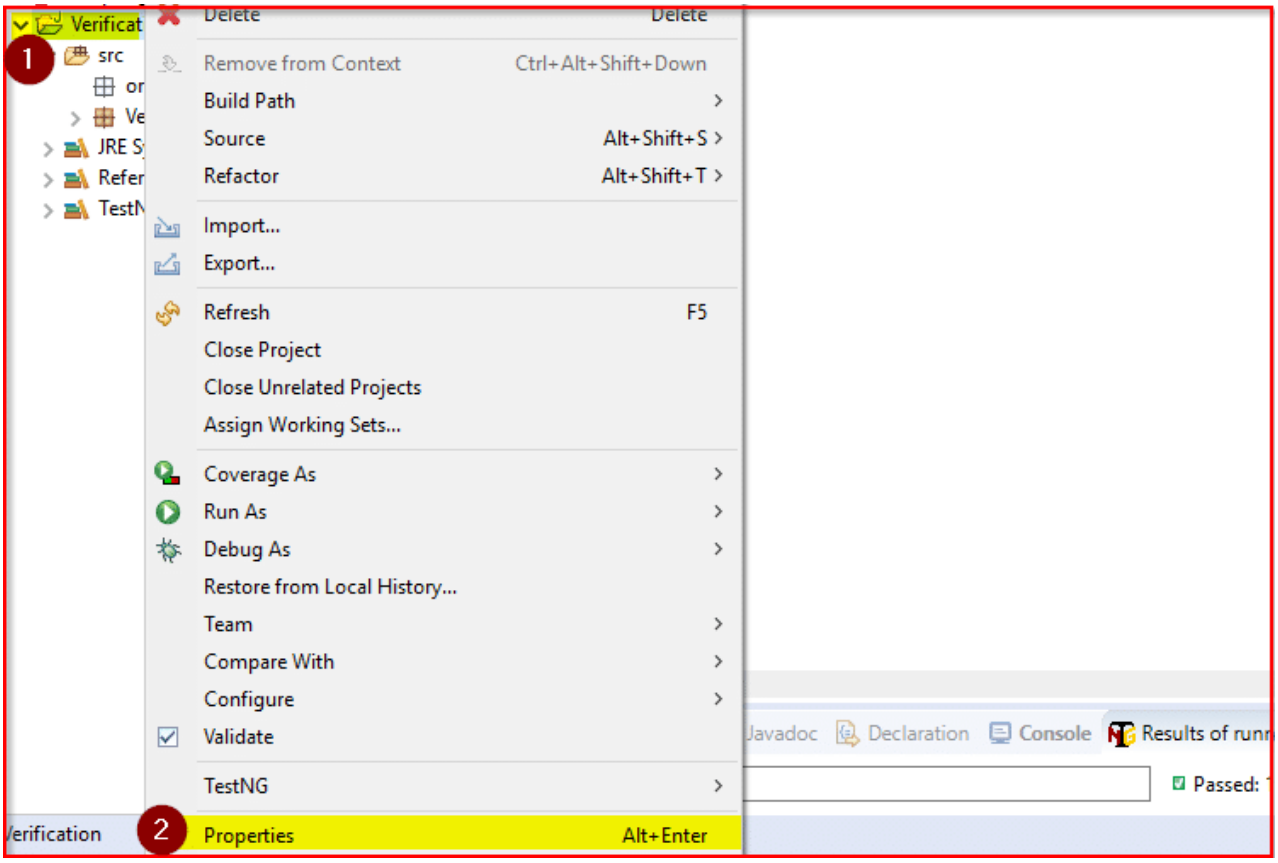


Step 3: When the below window pops up, click on '**Don't Create**,' for not creating a module for your project



Step 4:

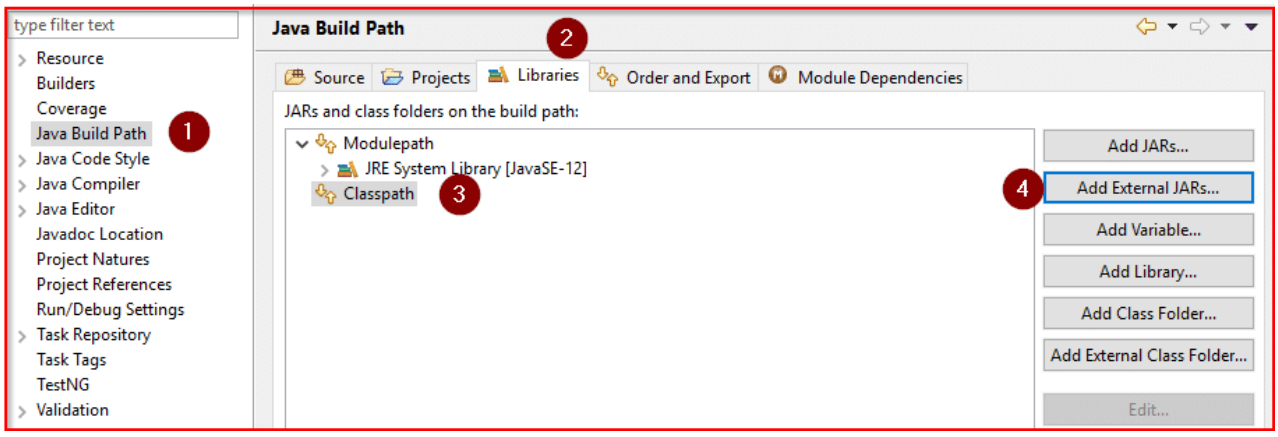
1. Right-click on your **project name**
2. Then, click on **Properties**



Step 5: The below-shown window will popup

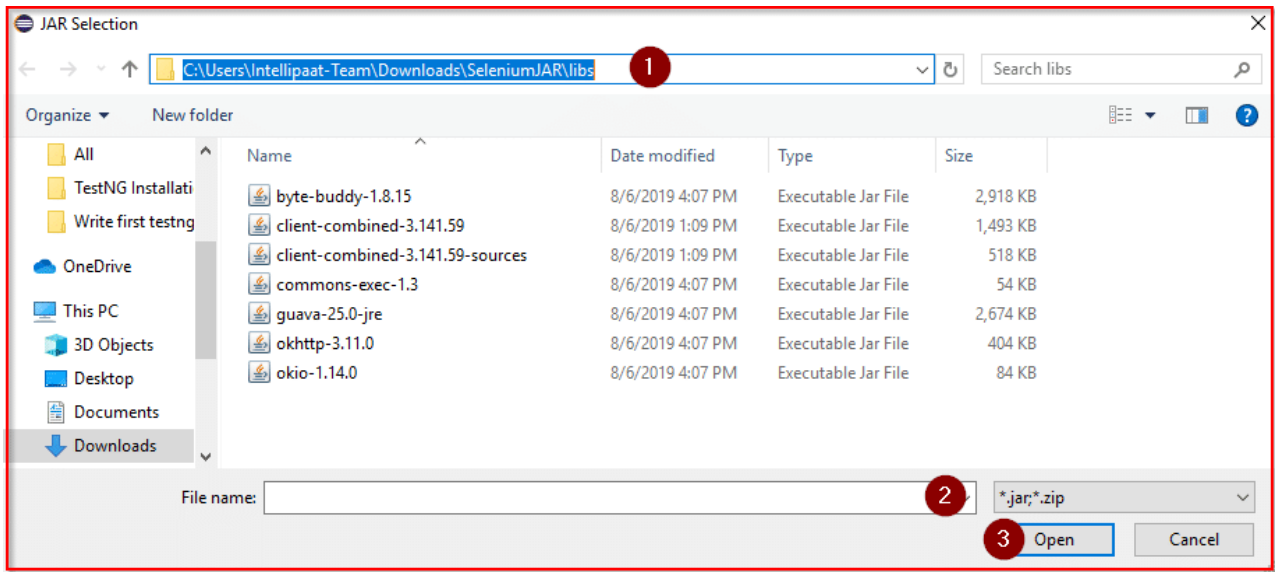
1. Click on **Java Build Path**
2. Click on **Libraries**

3. Select **Classpath**
4. Click on **Add External JARs**



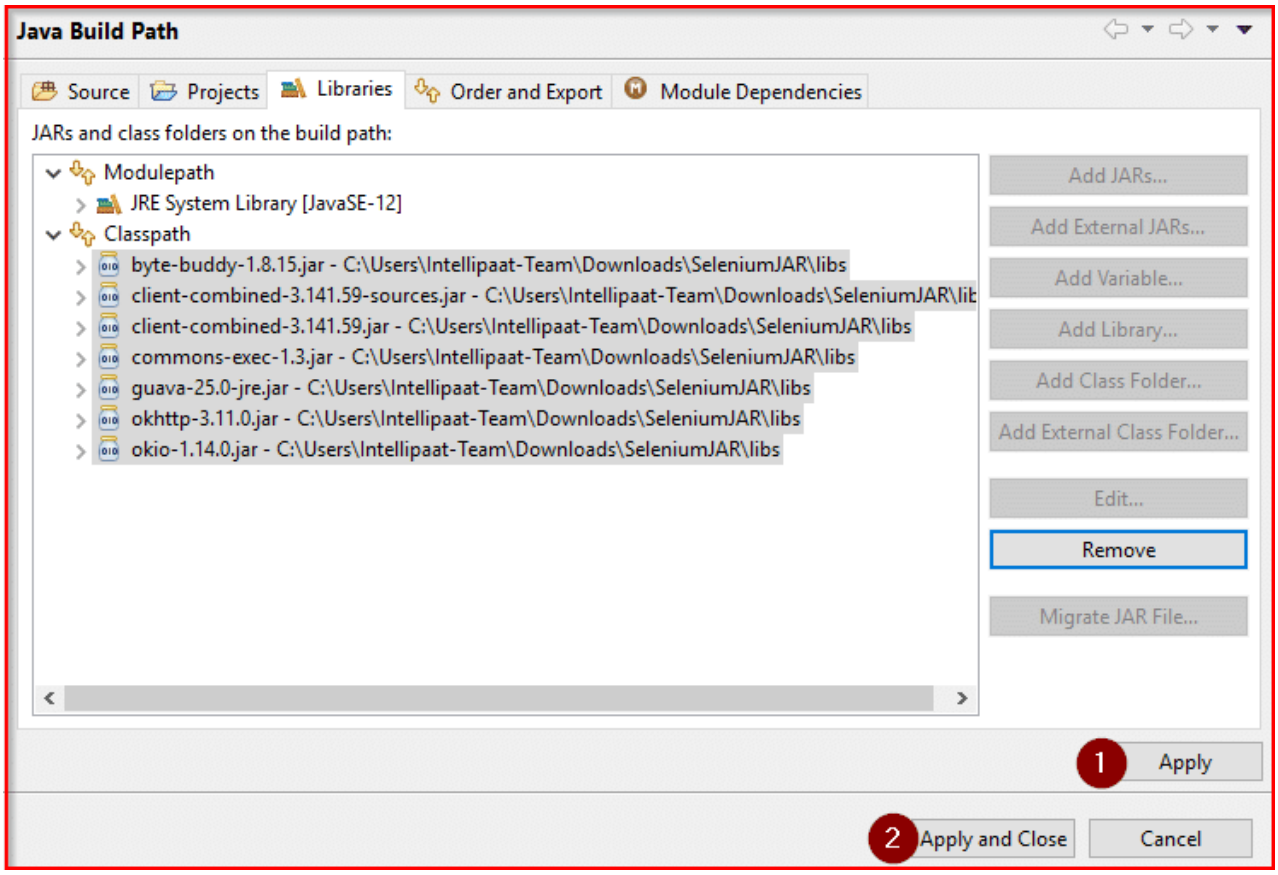
Step 6:

1. The below location will popup, wherein you will have all the **Selenium Client and WebDriver Language Bindings** JAR files which were extracted into a new folder from the **Download** menu of the www.seleniumhq.org site.
2. Press **Ctrl+A** to select all the files in that folder
3. Click on **Open**



Step 7: All the .jar files should be populated in the **Classpath**

1. Click on **Apply**
2. Then, click on **Apply and Close**



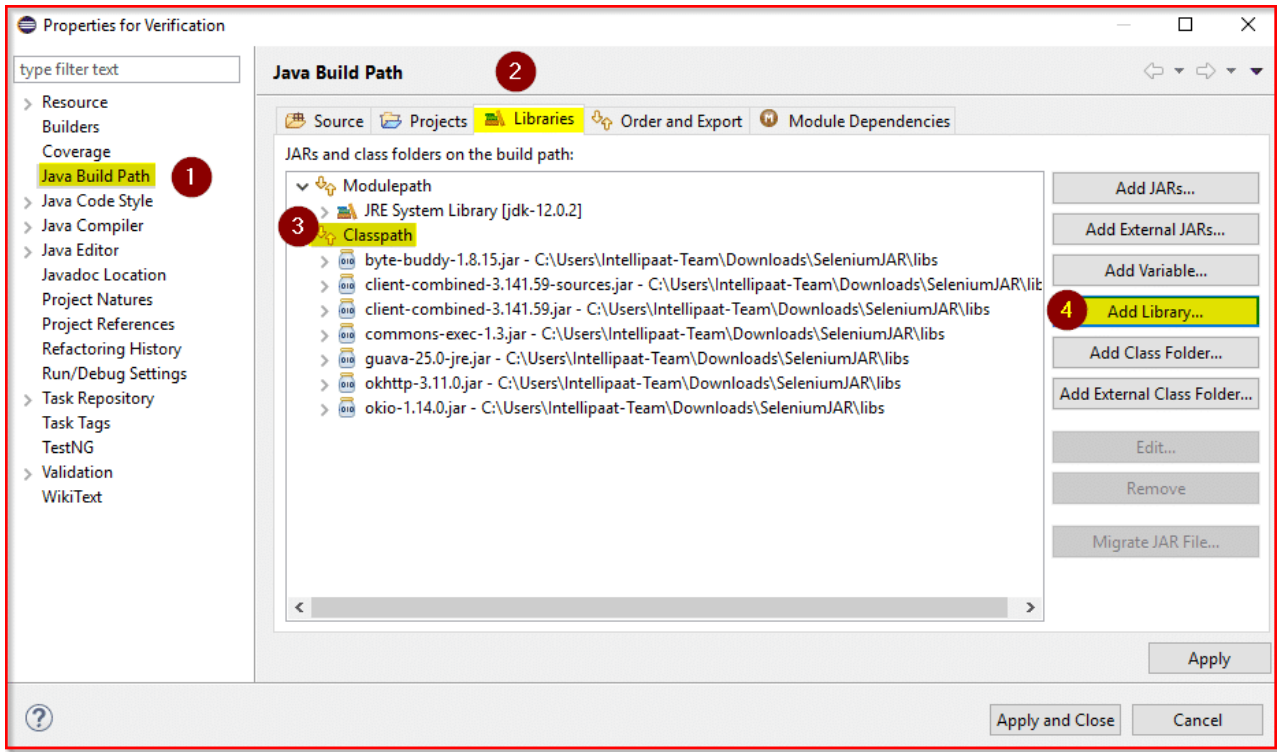
Under your **project folder** (here '**Verification**') in the **Package Explorer**, you will find the **Referenced Libraries** folder being created containing all those .jar files within it.

Step 8: Now, **add TestNG library** to your project.

Note: The addition of external jars and the addition of TestNG library need to be done for every new project you create.

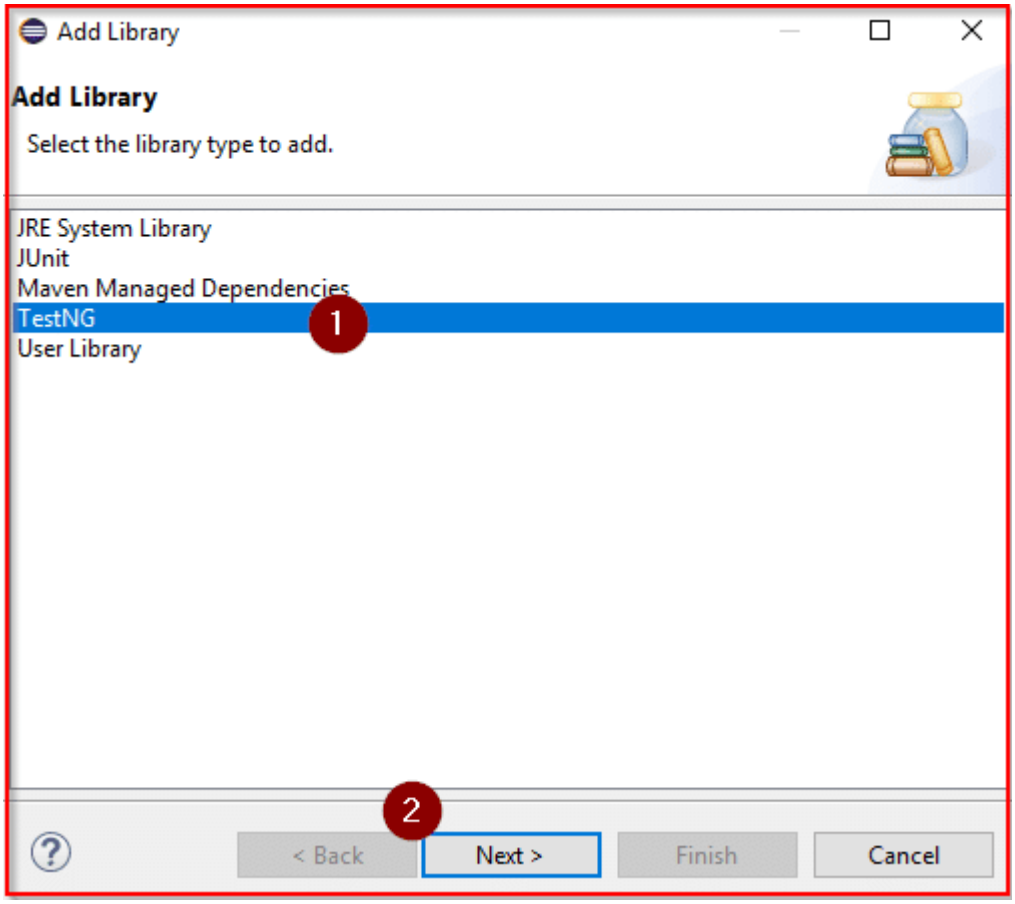
After right-clicking on your project name and navigating to properties,

1. Click on **Java Build Path**
2. Click on **Libraries**
3. Select **Classpath**
4. Click on **Add Library**



Step 9: The below-shown window will popup

1. Select **TestNG**
2. Click on **Add** and then on **Finish**



A new **folder** named **TestNG** will get added under your project folder.

Now, you have successfully installed the TestNG framework with the help of Eclipse IDE. Let's now write the first test case using Test annotations.

Get certified from this top Selenium course in Singapore today!

TestNG Annotations

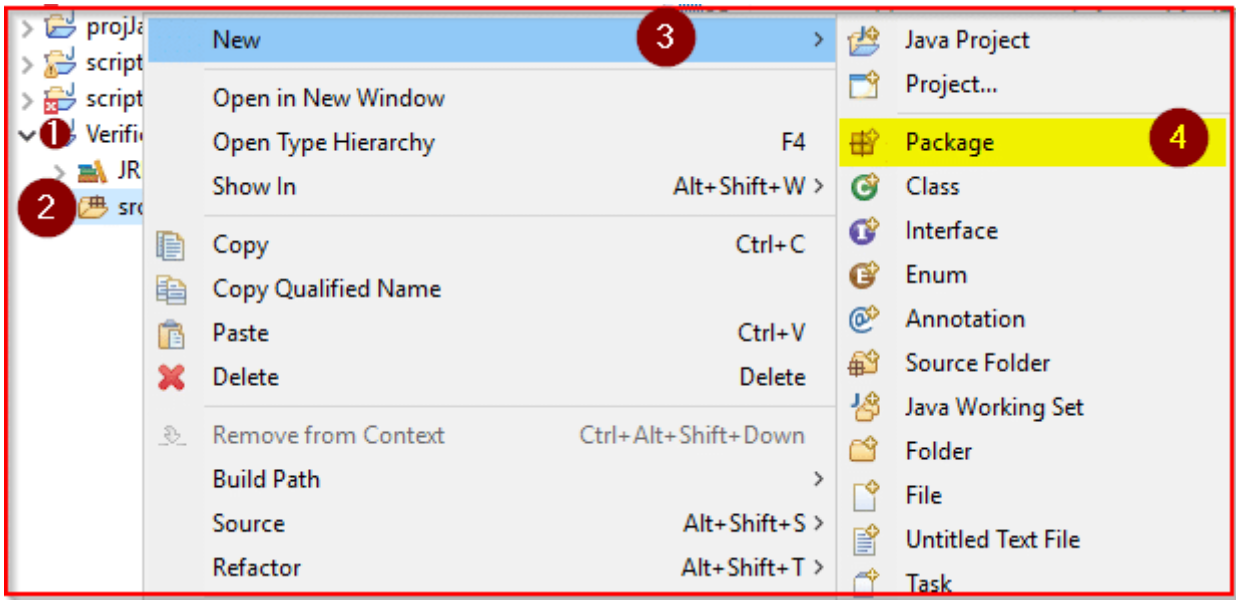
In this section of the TestNG in Selenium tutorial, you will be introduced to the @Test annotations, various other advanced annotations and their usages. Annotations differ from project to project depending on their necessities. Though the requirements change, the execution flow will be the same for every single project.

In this section of the tutorial, you will learn about various advanced TestNG annotations and their usages.

Here are the steps to perform web application automation testing by creating multiple test annotated method and passing multiple attributes/parameters in @Test annotations:

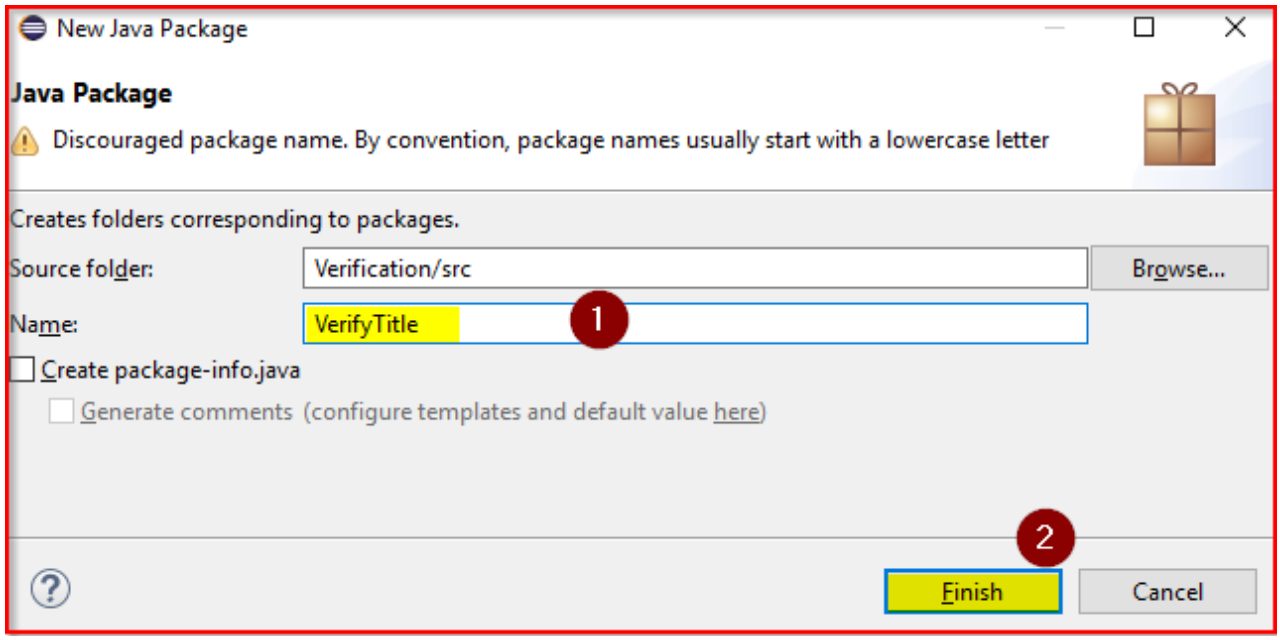
Step 1:

1. Click on your project name (here, '**Verification**')
2. Then, click on the **src** folder
3. Select **New**
4. Then, click on **Package**



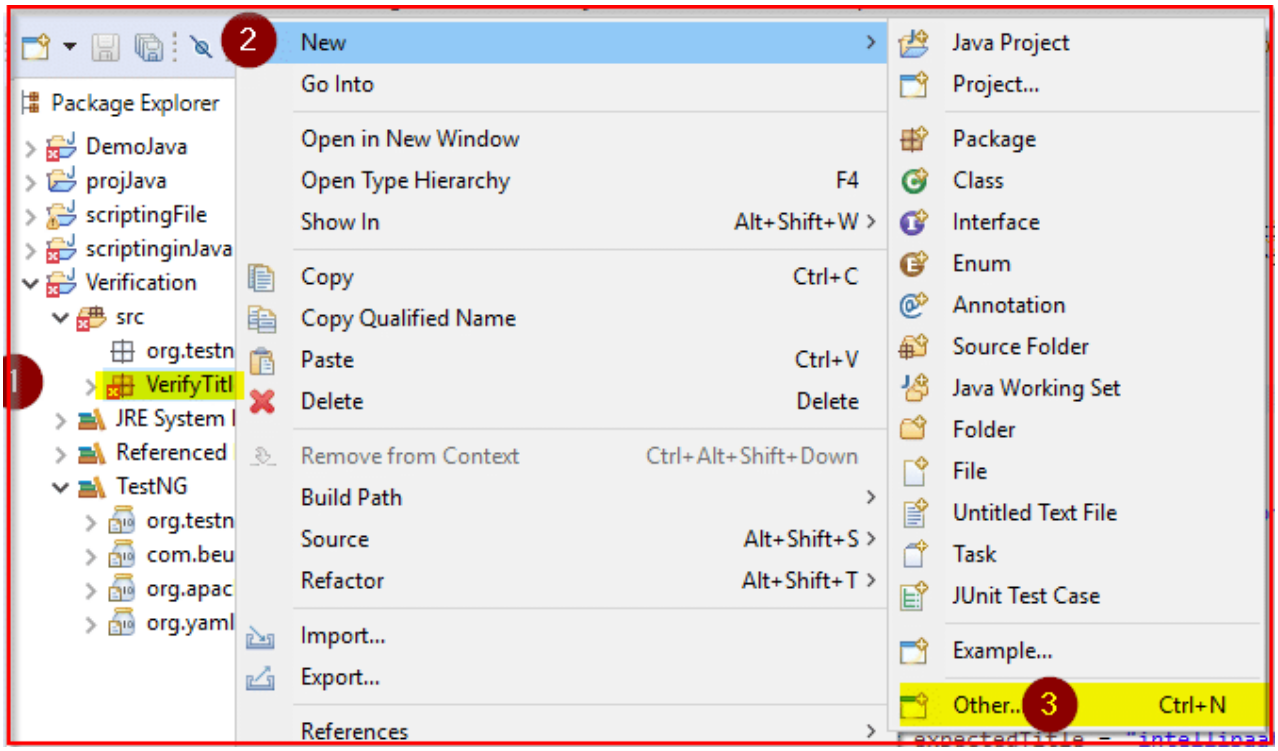
Step 2:

1. Enter the **Name** of the package, say **VerifyTitle**
2. Click on **Finish**



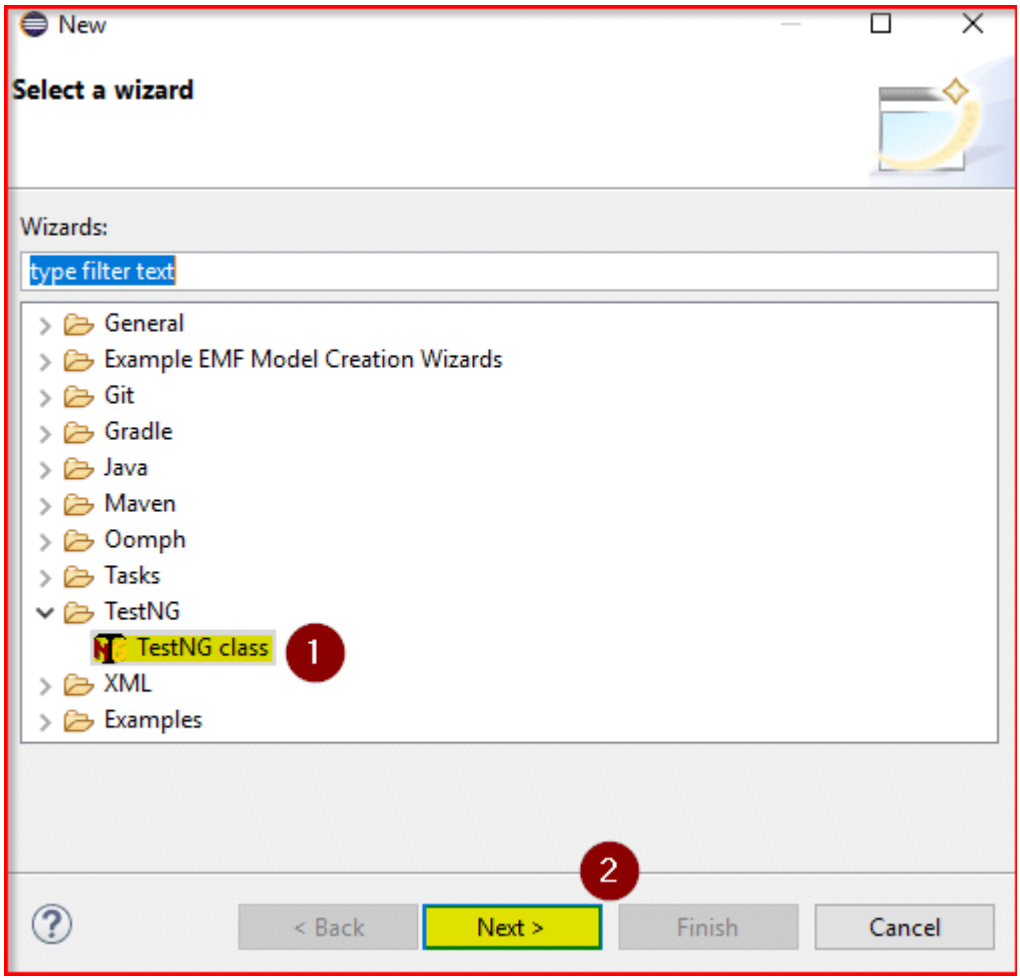
Step 3: Create a TestNG class

1. Right-click on your **package name** (here, **VerifyTitle**)
2. Click on **New**
3. Click on **Other**



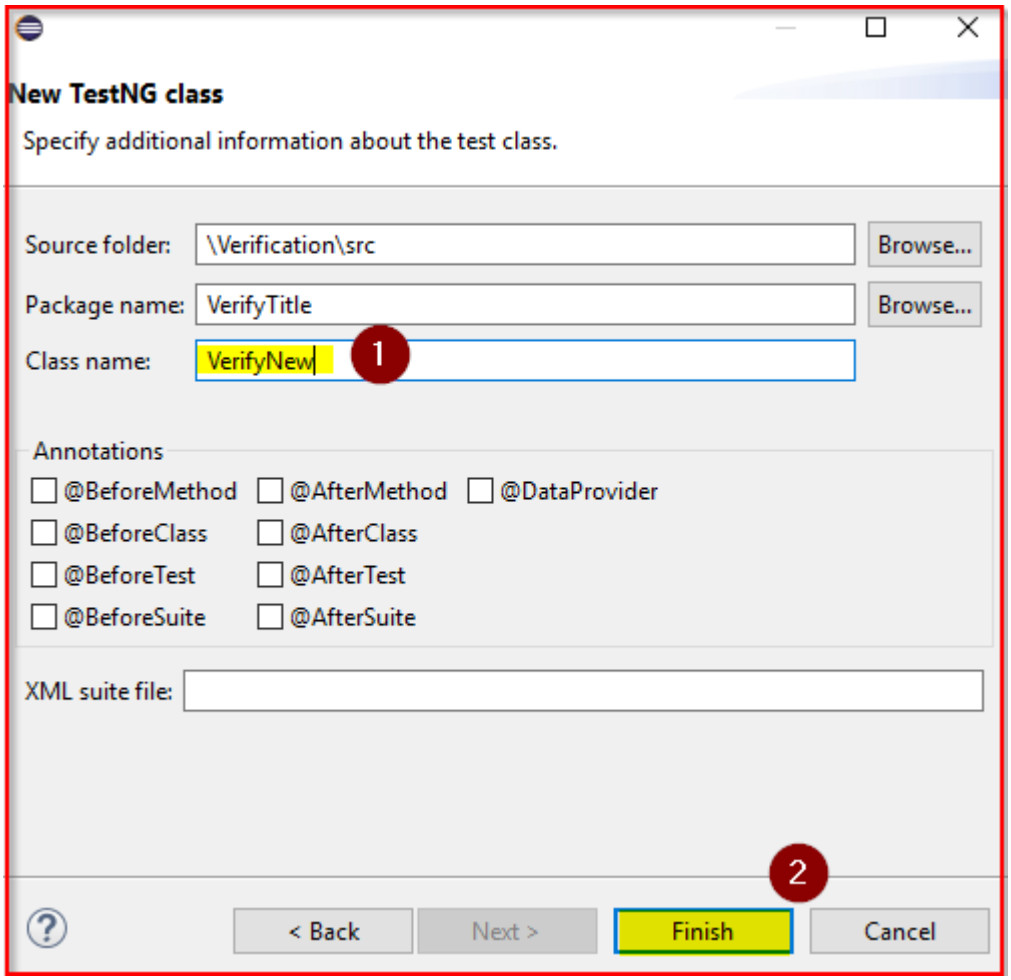
Step 4:

1. Click on **TestNG class**
2. Click on **Next**



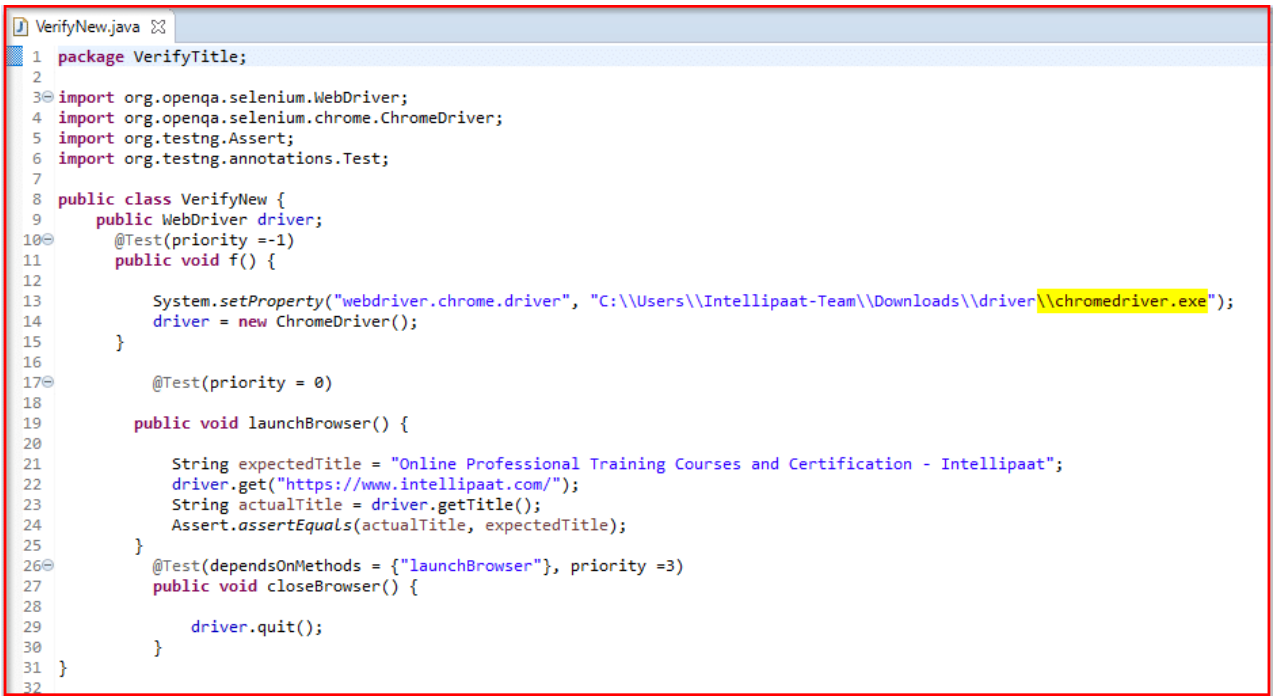
Step 5:

1. Enter your TestNG class name, say **VerifyNew**
2. Click on **Finish**



Step 6: Copy the below code and execute it in your local system's Eclipse IDE

```
package VerifyTitle;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.Test;
public class VerifyNew {
public WebDriver driver;
@Test(priority =-1)
public void f() {
System.setProperty("webdriver.chrome.driver", "C:\\Users\\Intellipaat-
Team\\Downloads\\driver\\chromedriver.exe");
driver = new ChromeDriver();
}
@Test(priority = 0)
public void launchBrowser() {
String expectedTitle = "Online Professional Training Courses and Certification -
Intellipaat";
driver.get("https://www.intellipaat.com/");
String actualTitle = driver.getTitle();
Assert.assertEquals(actualTitle, expectedTitle);
}

@Test(dependsOnMethods = {"launchBrowser"}, priority =3)

public void closeBrowser() {


driver.quit();

}

}
```



Code Explanation

Line 9: Global object (here, **driver**) declaration which is of type WebDriver; we need a global declaration of driver-object so that all the test-annotated methods can perform driver-driven actions efficiently

Note: If it shows an error, mouse-over the red line. A suggestions list appears, from which you have to import the TestNG WebDriver library, **import org.openqa.selenium.WebDriver;**

Line 10: A method named '**f**' is annotated under the test annotation with priority set to -1, will be run first.

Note: If you see a red line as a sign of error, mouse-over it and import the **import org.testng.annotations.Test;** test annotation library.

Line 13: The **System.setProperty(key,value)** method, sets the system property key (here, **webdriver.chrome.driver**) to have the value,

```
C:\\Users\\Intellipaat-Team\\Downloads\\driver\\chromedriver.exe
```

In Selenium, you use this method because the browser doesn't have a built-in server to run the automation code, so you will need a Chrome/IE/Gecko (according to your requirement) driver server for communicating your **Selenium** code to the browser.

Note: At the end of the path location of the WebDriver, in the value parameter, make sure to add **(path...\\chromedriver.exe");**

Line 14: By using this, your scripts are now flexible and you can use any WebDriver object (here, driver) which is required to invoke a particular browser (here, Chrome Browser). The reference variable (driver) of type WebDriver class allows us to assign the driver object to different browser-specific drivers.

Note: If you see a red line as a sign of error, mouse hover it and import the library

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

Line 19: You have defined a second test annotated method named launchBrowser(), with a priority set to 0, which will be run after the f() method has run.

Line 21: A variable named **expectedTitle** of the data type string holds a string value which is the title of the website to be visited

Line 22:

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

is used to navigate to a URL provided by you and waits until the page is fully loaded.

Line 23: The title of the site, retrieved by using the **driver.getTitle()**method, is stored in a string data type variable, named **actualTitle**

Line 24:

```
Assert.assertEquals(actual, expected)
```

will compare the actual value in the first parameter to the expected value in the second parameter and will show the result in the console window, to check whether all the test annotated methods passed or not.

Note: If you see a red line as a sign of error, mouse-over it and import the required library **import org.testng.Assert;**

Line 26: You are using **dependsOnMethod** attribute within @Test annotation; we can specify the name of the parent test method on which the test should be dependent, here on method **launchBrowser()**

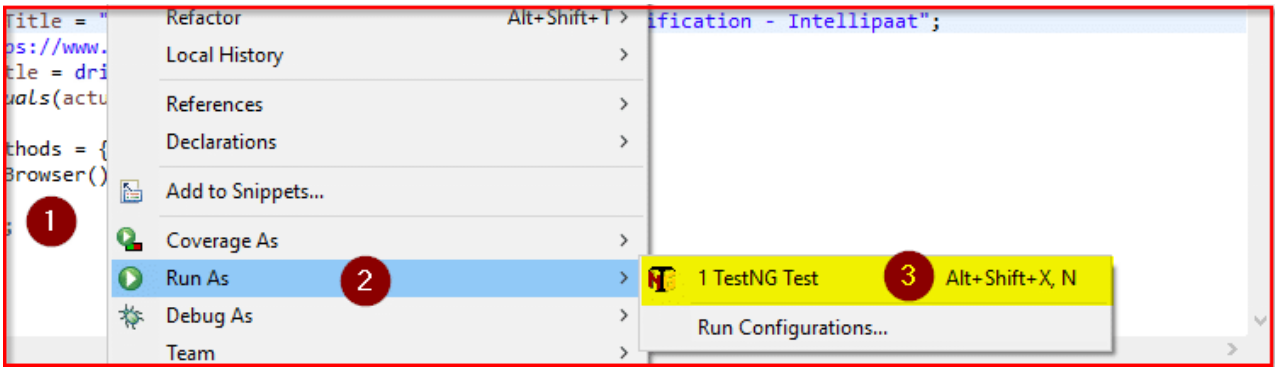
Line 27: You have defined a third test annotated method named **closeBrowser(),** with a priority set to 3, which will be run after the **launchBrowser()**method has run.

Line 29: **driver.quit()** will close all the browser windows and end the WebDriver session gracefully.

Become a master of TestNG in Selenium by going through this online Selenium training in Toronto!

Step 7: To run your TestNG test case, save it by pressing Ctrl+S and then

1. Right-click on the coding window
2. Select **Run As**
3. Click on **1 TestNG Test**



Step 8: Take a look at the console window. This shows that the methods ran as per the priority assigned to each of the test annotated methods and not in alphabetical order (the default way)

First, the method ‘f’ got executed, then ‘launchBrowser’, and then ‘closeBrowser’

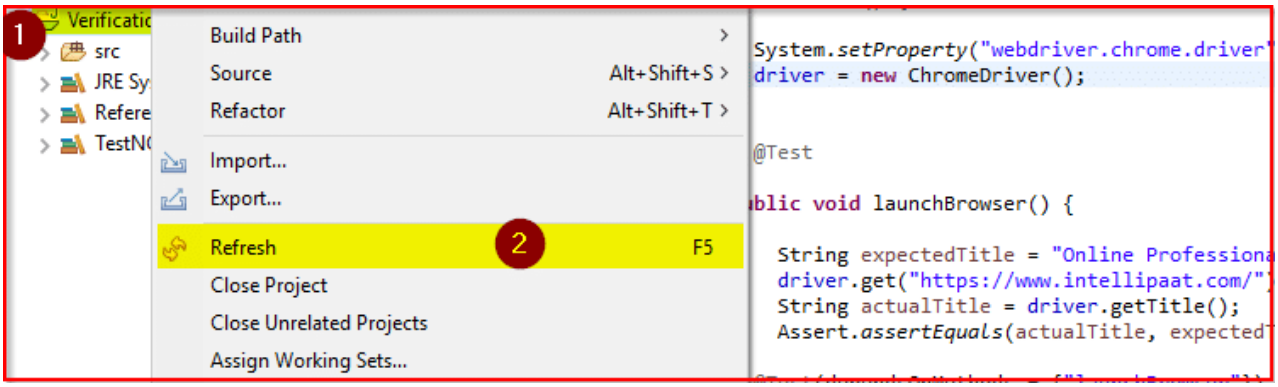
```
PASSED: f
PASSED: launchBrowser
PASSED: closeBrowser

===============================================
    Default test
    Tests run: 3, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 3, Failures: 0, Skips: 0
===============================================
```

View the HTML Test Report

In this TestNG in Selenium tutorial, till now you must have understood how to perform web application automation testing by creating multiple test annotated method and passing multiple attributes as a parameter in the @Test annotations. Let us move ahead and see how to view the html test report.

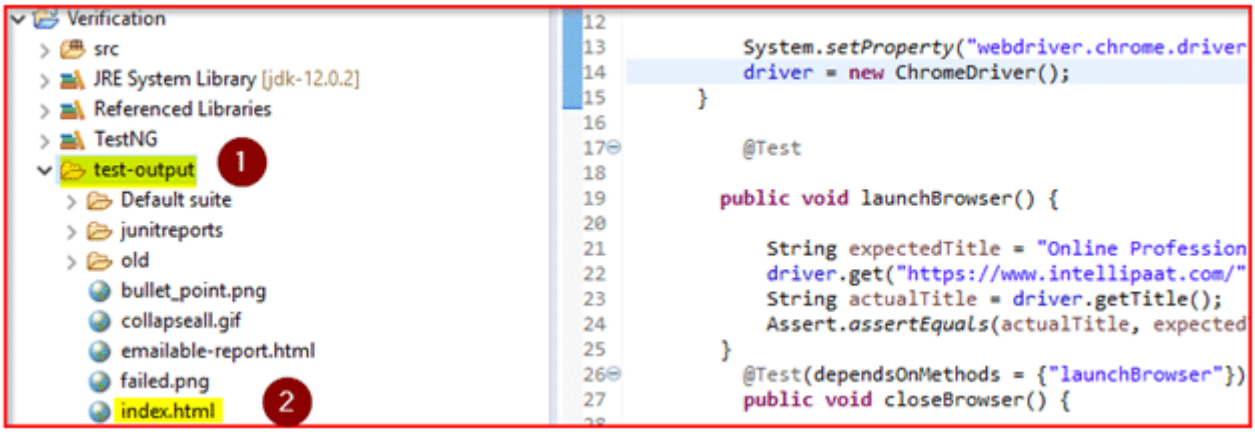
Step 1: (1) Right-click on your project name folder (**Verification**)

(2) Click on **Refresh**

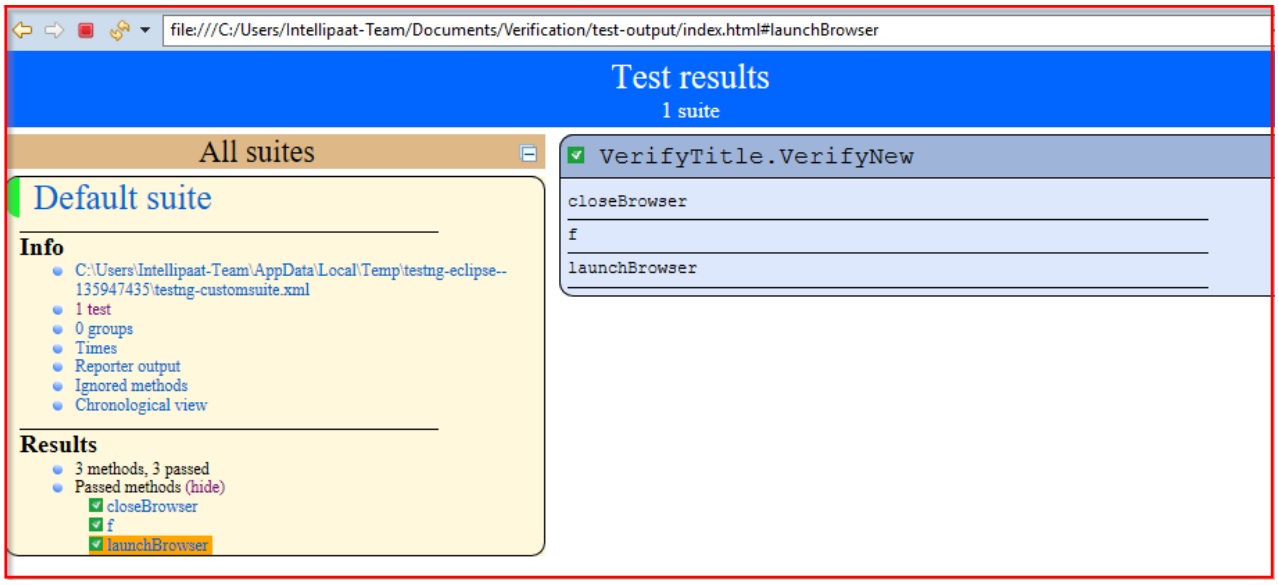


Step 2: A new test-output folder will be added

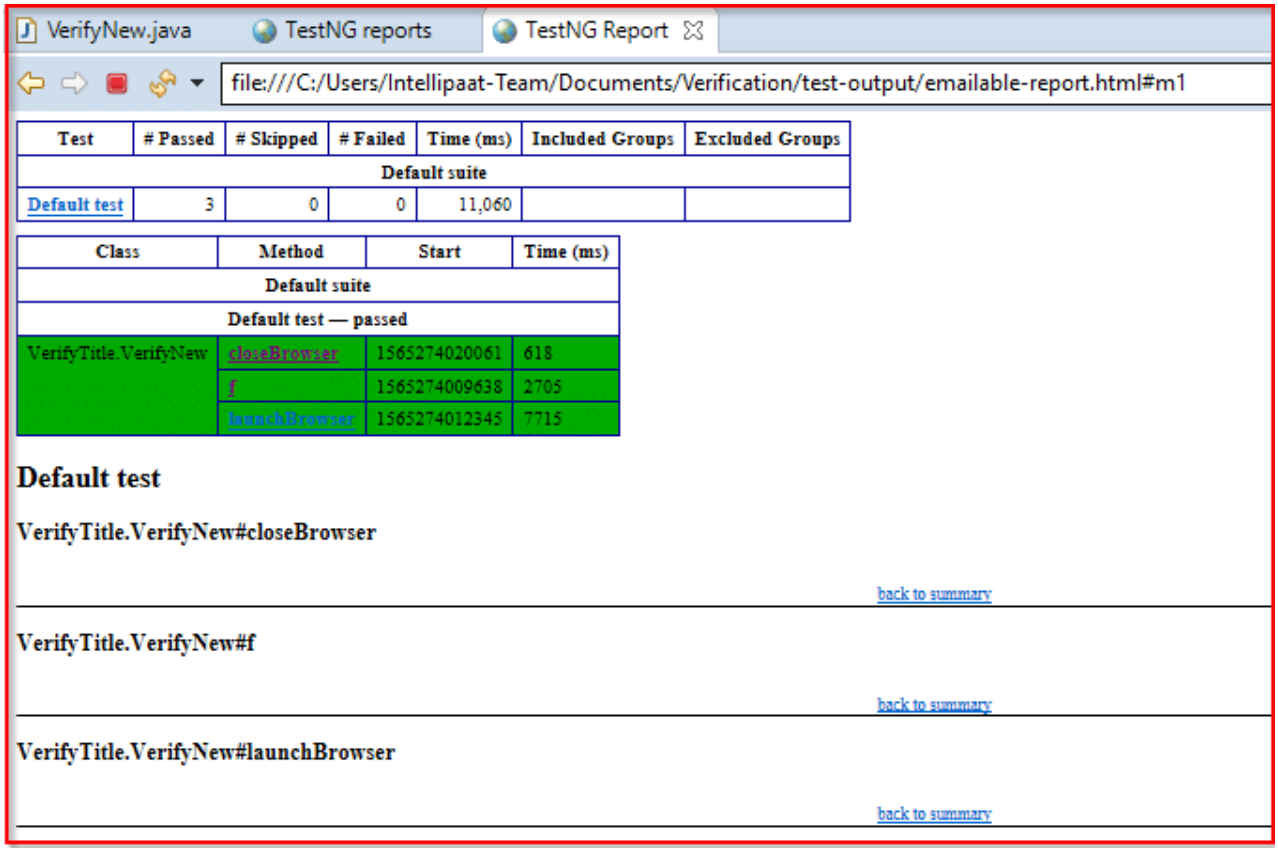
1. Click on **test-output**
2. Double-click on **index.html**



Step 3: An **html test report** is generated, which looks like the image shown below:

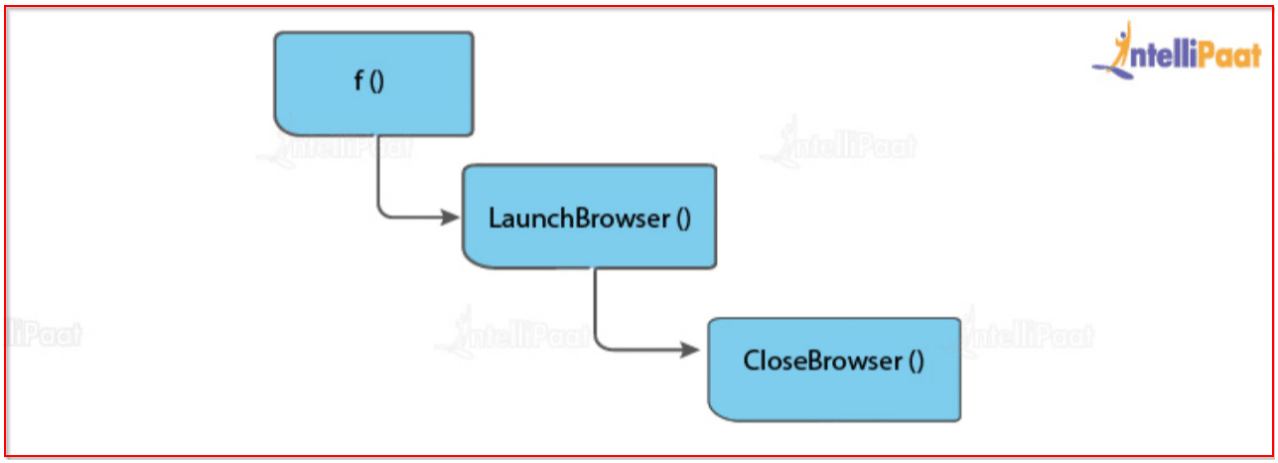


Step 4: You can also view the **emailable test report** by double-clicking on the emailable-report.html under test-output folder, which would look like the image shown below:



Test Execution Flow:

1. First, f()launches the browser.
2. Then, launchBrowser() verifies the title and matches the actual title with the expected title.
3. After that, closeBrowser() closes the browser.



Note:

f(): Launching of the browser is a **precondition for every test case**.

closeBrowser(): It is a **postcondition for every test case**.

Hence, we have two annotations which serve the same purpose, and they are @BeforeMethod and @AfterMethod.

- **@BeforeMethod**: The annotated method will be run before each test method. A method annotated with @BeforeMethod annotation is a good place to initialize some setup which can be used and eventually altered by the @Test annotated method.
- **@AfterMethod**: The annotated method will get executed after each test method. The @AfterMethod annotated method is a handy place to clean up the setup created (like the initialization of the browser) in the @BeforeMethod and updated by the @Test method.

VerifyNew.java

```
package VerifyTitle;

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 VerifyNew {
    public WebDriver driver;
    @BeforeMethod
    public void launchBrowser() {
        System.setProperty("webdriver.chrome.driver", "C:\\Users\\Intellipaat-Team\\Downloads\\driver\\chromedriver.exe");
        driver= new ChromeDriver();
    }

    @Test(priority=2)
    public void verifyTitle1() {
        driver.get("https://www.intellipaat.com/");
        Assert.assertEquals("Online Professional Training Courses and Certification - Intellipaat", driver.getTitle());
    }

    @Test(priority=1)
    public void verifyTitle2() {
        driver.get("https://www.google.com/");
        Assert.assertEquals("Google", driver.getTitle());
    }

    @AfterMethod
    public void  closeBrowser() {
        driver.close();
    }
}
```

(1) verifyTitle2() with priority 1 will be run first.
(2) verifyTitle1() with priority 2 will be run second.

Code Explanation

Line 11: Global object (here, **driver**) declaration which is of type WebDriver class; we need a global declaration of driver-object so that all the test-annotated methods can perform driver-driven actions efficiently.

Note: If it shows an error, mouse-over the red line. A suggestions list appears, from which you have to import the TestNG WebDriver library, **import org.openqa.selenium.WebDriver;**

Line 13: A method **launchBrowser()** defined under **@BeforeMethod** annotation**.** The method **launchBrowser**() annotated with @BeforeMethod annotation is a good place to initialize some setup which can be used and eventually altered by the @Test annotated method.

Note: If you see a red line as a sign of error, mouse-over it and import the **import org.testng.annotations.BeforeMethod;** test annotation library.

Line 14: The **System.setProperty(key,value)** method, sets the system property key (here, **webdriver.chrome.driver**) to have the value,

```
C:\\Users\\Intellipaat-Team\\Downloads\\driver\\chromedriver.exe
```

In Selenium, you use this method because the browser doesn't have a built-in server to run the automation code, so you will need a Chrome/IE/Gecko (according to your requirement) driver server for communicating your **Selenium** code to the browser.

Note: At the end of the path location of the WebDriver, in the value parameter, make sure to add **(path...\\chromedriver.exe");**

Line 19: A method named **verifyTitle1()** is annotated under the test annotation with priority set to 2.

Note: If you see a red line as a sign of error, mouse-over it and import the **import org.testng.annotations.Test;** test annotation library.

Line 20: **driver.get("https://www.intellipaat.com/")**used to navigate to a URL provided by you and waits until the page is fully loaded.

Line 21:

```
Assert.assertEquals(("Online Professional Training Courses and and Certificates - Intellipaat",driver.getTitle())
```

matches the actual title(string passed in the first parameter) with the expected title(string received at the run time using **driver.getTitle()**in the second parameter).

Note: If you see a red line as a sign of error, mouse-over it and import the required library **import org.testng.Assert;**

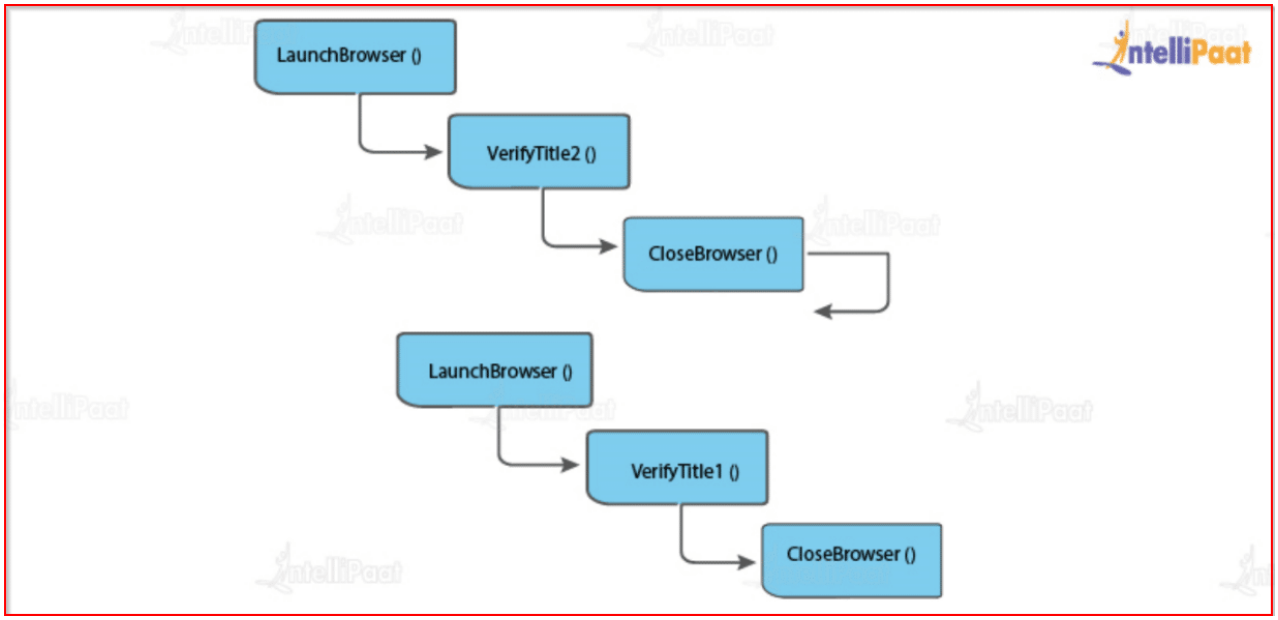
Line 25: A method named **verifyTitle2()** is annotated under the test annotation with priority set to 1.

Note: Do make a note that by default the TestNG test cases are run in alphabetical order, to control the flow of test cases we used priority attribute, here, method verifyTitle2() with priority equal to 1 will be executed before verifyTitle1() whose priority was 2.

Line 31: A method **closeBrowser()** defined under **@AfterMethod** annotation**.** The method **closeBrowser**() annotated with @AfterMethod is a good place to clean up the setup created (like the initialization of the browser) and used to close the chrome browser currently in focus.

Learn end-to-end TestNG in Selenium concepts through the Selenium training in Hyderabad to take your career to a whole new level!

Test Execution Flow:



Result in Console Window:

```
PASSED: verifyTitle2
PASSED: verifyTitle1

===============================================
    Default test
    Tests run: 2, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 2, Failures: 0, Skips: 0
===============================================
```

Note: It shows only two test cases in the console, but the execution flow is of six steps i.e. launchBrowser() -> verifyTitle2()-> closeBrowser()-> launchBrowser()-> verifyTitle1()-> closeBrowser()

Now, let's see the implementation of the other two test annotations which are @BeforeClass and @AfterClass.

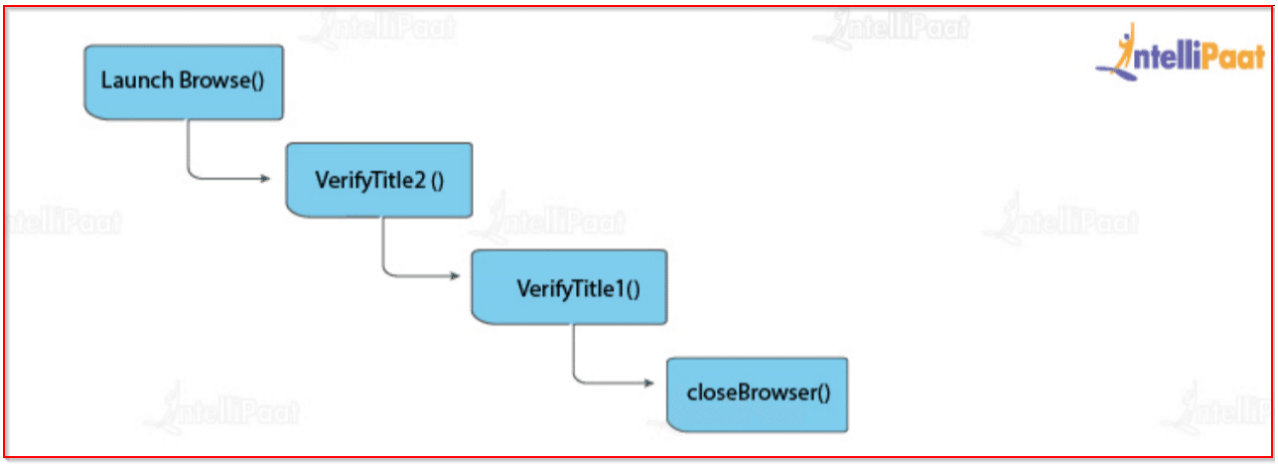
1. **@BeforeClass**: The annotated method that runs before the **first test method** in the current class is **invoked**. It is a good place to do any initialization or configuration setup which is common to all test methods and eventually may be used in those test methods.
2. **@AfterClass**: The annotated method will be run after all the test methods in the current class have been run. It provides a handle to do some cleanups after all tests in the current class have been executed.

```
1  package VerifyTitle;
2  import org.openqa.selenium.WebDriver;
10
11 public class VerifyNew {
12     public WebDriver driver;
13       @BeforeClass
14       public void launchBrowser() {
15           System.setProperty("webdriver.chrome.driver", "C:\\Users\\Intellipaat-Team\\Downloads\\driver\\chromedriver.exe");
16           driver= new ChromeDriver();
17           System.out.println("@BeforeClass annotated method runs before the first test method in the current class is invoked.");
18       }
19
20       @Test(priority=2)
21       public void verifyTitle1() {
22           driver.get("https://www.intellipaat.com/");
23           Assert.assertEquals("Online Professional Training Courses and Certification - Intellipaat", driver.getTitle());
24           System.out.println("Method verifyTitle1() executed!");
25       }
26
27       @Test(priority=1)
28       public void verifyTitle2() {
29           driver.get("https://www.google.com/");
30           Assert.assertEquals("Google", driver.getTitle());
31           System.out.println("Method verifyTitle2() executed!");
32         }
33
34       @AfterClass
35       public void  closeBrowser() {
36           System.out.println("@AfterClass annotated method is run after all the test methods in the current class have been run!");
37           driver.close();
38               }
39       }
```

The result in the console window looks like this:

```
@BeforeClass annotated method runs before the first test method in the current class is invoked.
Method verifyTitle2() executed!
Method verifyTitle1() executed!
@AfterClass annotated method is run after all the test methods in the current class have been run!
PASSED: verifyTitle2
PASSED: verifyTitle1

===============================================
    Default test
    Tests run: 2, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 2, Failures: 0, Skips: 0
===============================================
```

Test Execution Flow:



Let us advance in the TestNG in Selenium tutorial and see the other set of test annotations @BeforeTest and @AfterTest:

@BeforeTest: The annotated method will be run before any test method present in all those classes mentioned inside the <test> tag is run.

@AfterTest: The annotated method will be run after all the test methods present in all those classes which are inside the <test> tag in the .xml file have run.

```
package VerifyTitle;
import org.openqa.selenium.WebDriver;

public class VerifyNew {
    public WebDriver driver;
      @BeforeTest
      public void launchBrowser() {
          System.setProperty("webdriver.chrome.driver", "C:\\Users\\Intellipaat-Team\\Downloads\\driver\\chromedriver.exe");
          driver= new ChromeDriver();
          System.out.println("Method launchBrowser() executed!");
      }

      @Test(priority=2)
      public void verifyTitle1() {
          driver.get("https://www.intellipaat.com/");
          Assert.assertEquals("Online Professional Training Courses and Certification - Intellipaat", driver.getTitle());
          System.out.println("Method verifyTitle1() executed!");
      }

      @Test(priority=1)
      public void verifyTitle2() {
          driver.get("https://www.google.com/");
          Assert.assertEquals("Google", driver.getTitle());
          System.out.println("Method verifyTitle2() executed!");
        }

      @AfterTest
      public void  closeBrowser() {
          System.out.println("Method closeBrowser() executed!");
          driver.close();
              }
      }
```

The result in the console window looks like this:

```
Method launchBrowser() executed!
Method verifyTitle2() executed!
Method verifyTitle1() executed!
Method closeBrowser() executed!
PASSED: verifyTitle2
PASSED: verifyTitle1

===============================================
    Default test
    Tests run: 2, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 2, Failures: 0, Skips: 0
===============================================
```

Note the difference between @BeforeTest and @BeforeMethod:

- @BeforeTest: It will be called only once before any test methods, no matter how many test-annotated methods are present within the class.
- @BeforeMethod: It will be invoked before every test method; if you have 10 test-annotated methods, it will be called 10 times.

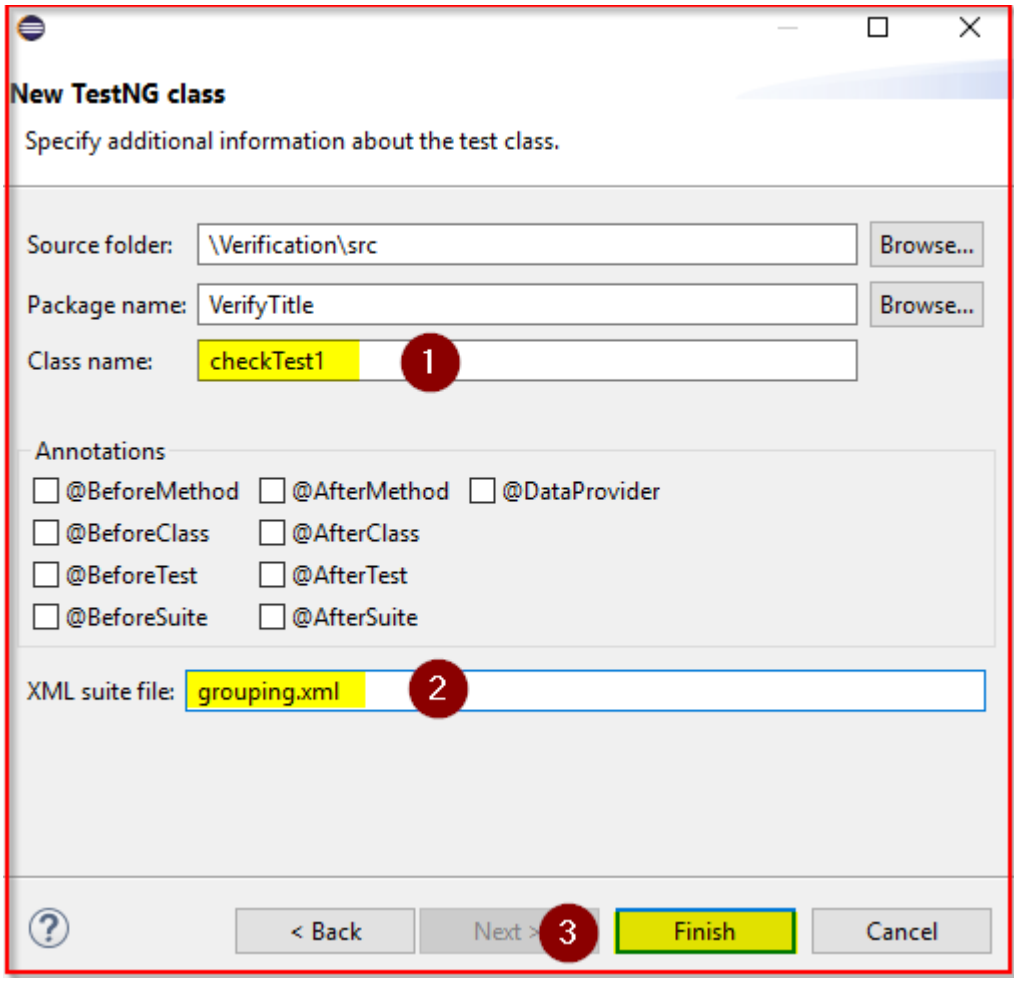
Go through this Selenium course in London to get a clear understanding of TestNG in Selenium!

Group Multiple Test Cases

TestNG in Selenium allows us to group several tests. We can group certain tests based on what behaviour/aspect they are testing. We may have a scenario where few tests belong to a certain group (say, sanity) and others belong to another group (say, regression), and yet another one belonging to yet another group (say, functional). With this approach, we may decide to execute only a certain group of tests and skip other ones (only the sanity-related code is required there, so it is preferred to execute only the sanity group-related tests). Several tests may belong to a group, and a test can be included in multiple groups. Let's begin.
Step 1: Create an XML file in the same package by right-clicking on the **package name**

Step 2: Click on **NewStep 3:** Click on **OtherStep 4:** Select **TestNG** and then **FinishStep 5:**

1. Type the **Class name** (say, **checkTest1**)
2. Type the **XML suite file** name (say, **grouping.xml**)
3. Click on **Finish**



Step 6: On the coding window, on the bottom-left side, click on **Source** instead of **Design** and add three more tags as shown in the image below:

Step 7: Copy and write the below code in your local system's Eclipse IDE, which depicts the implementation of @BeforeGroups and @AfterGroups annotations

```
package VerifyTitle;

import org.testng.annotations.AfterGroups;

import org.testng.annotations.BeforeGroups;
```

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


public class checkTest1 {


@BeforeGroups(groups= {"sanity"})

public void login() {

System.out.println("Login Successful.");

}


@Test(groups= {"sanity", "regression"}, priority = -1)

public void search() {

System.out.println("Search Value Entered.");

}


@Test(groups= {"sanity"}, priority = 0)

public void searchFromDropdown() {

System.out.println("Search from dropdown chosen
successfully.");

}


@Test(groups= {"regression"}, priority = 1)

public void courseSelect() {

System.out.println("Specific Course Selected.");

}


@AfterGroups(groups = {"sanity"})

public void logout() {

System.out.println("Logged out successfully.");

}

}
```

grouping.xml *checkTest1.java

```
1  package VerifyTitle;
2  import org.testng.annotations.AfterGroups;
3  import org.testng.annotations.BeforeGroups;
4  import org.testng.annotations.Test;
5
6  public class checkTest1 {
7
8    @BeforeGroups(groups= {"sanity"})
9    public void login() {
10       System.out.println("Login Successful.");
11   }
12
13   @Test(groups= {"sanity", "regression"}, priority = -1)
14   public void search() {
15       System.out.println("Search Value Entered.");
16   }
17
18   @Test(groups= {"sanity"}, priority = 0)
19   public void searchFromDropdown() {
20       System.out.println("Search from dropdown chosen succesfully.");
21   }
22
23   @Test(groups= {"regression"}, priority = 1)
24   public void courseSelect() {
25       System.out.println("Specific Course Selected.");
26   }
27
28   @AfterGroups(groups = {"sanity"})
29   public void logout() {
30       System.out.println("Logged out successfully.");
31   }
32 }
33
```

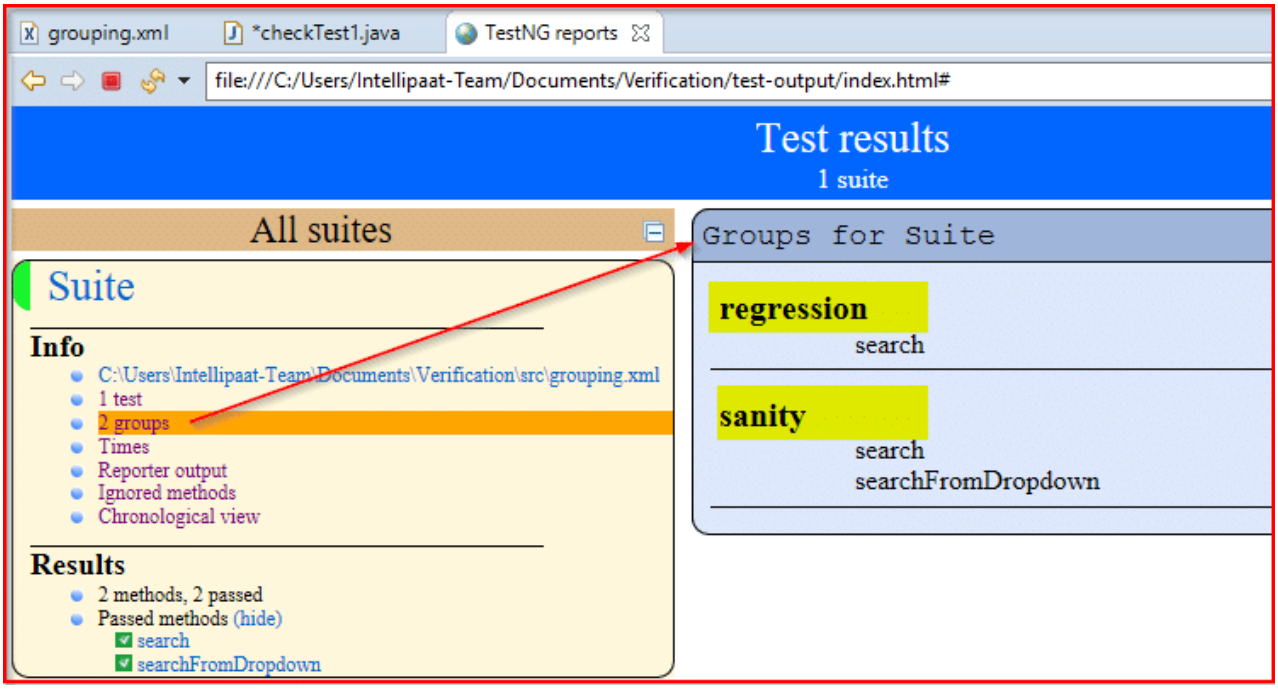
Step 8: Right-click on the **grouping.xml file** (because here, we are executing test cases using XML file), navigate to **Run As**, click on **1 TestNG suite**. You will see the following result in the console window

Problems Javadoc Declaration Console Results of running suite

```
<terminated> Verification_src.grouping.xml [TestNG] C:\Program Files\Java\jdk-12.0.2\bin\javaw.exe (Aug 9, 2019, 3:39:25 PM)
[RemoteTestNG] detected TestNG version 6.14.3
Login Successful.
Search Value Entered.
Search from dropdown chosen succesfully.
Logged out successfully.

===============================================
Suite
Total tests run: 2, Failures: 0, Skips: 0
===============================================
```

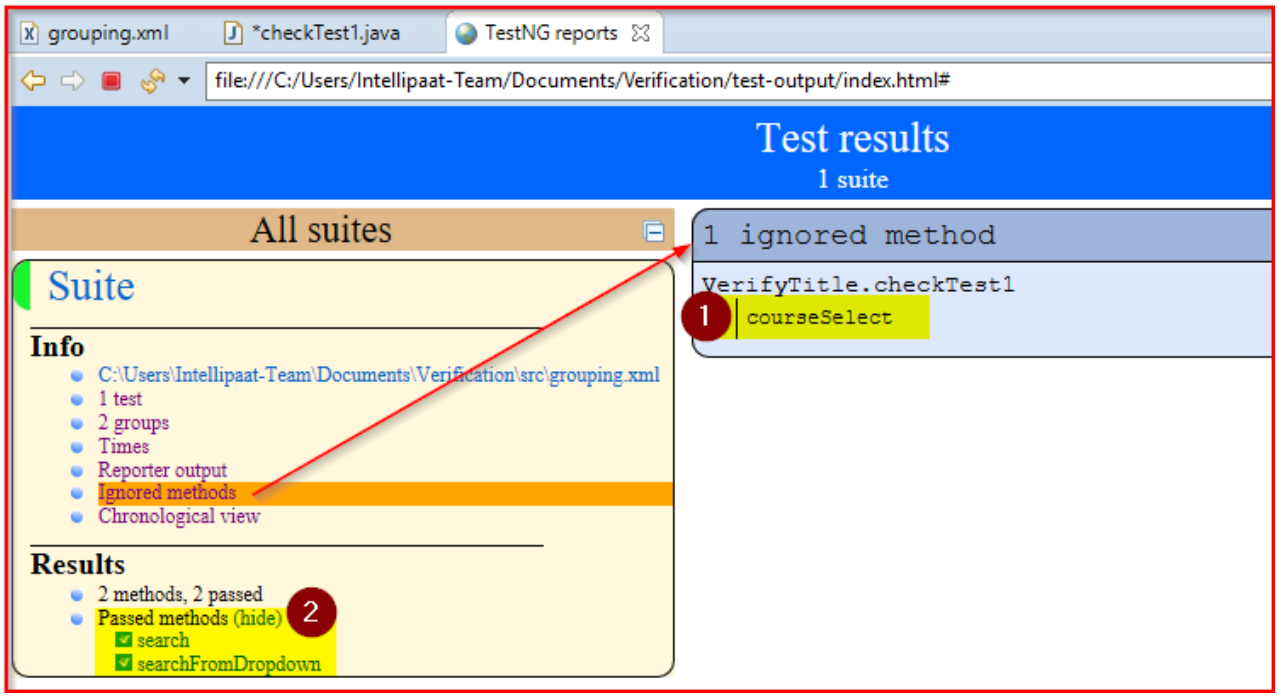
Note: It shows only two tests run in the console, but we wrote three test-annotated methods and it did not show any error. Why? Because as the group value mentioned in the parameter of the @Test annotation for the courseSelect() test annotated method is **regression** which is not the one **included** in the **<include>** tag within the **<groups>** tag of its **.xml file** (which is **sanity**), and hence TestNG framework doesn't identify this method to be a part of the 'sanity' group. Thus, it ignores and does not include it for testing, and further, the successive method gets executed readily.

Step 9: Click on **2 groups**. We will see **regression** and **sanity** groups and the methods defined under their names



Step 10:

1. Click on **Ignored methods** to see which method gets ignored. Here, the courseSelect() method gets ignored.The ignored method is **courseSelect()**.
2. Click on the **show** text to see the methods' names that passed the execution, here **search()** and **searchFromDropdown()**



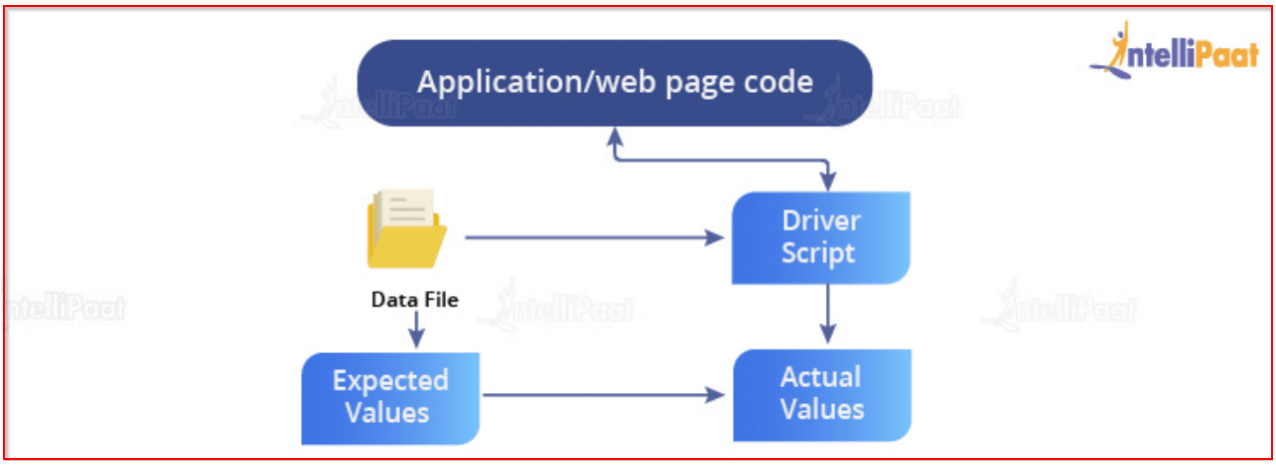
Visit our TestNG Community to get answers to all your queries!

In this TestNG in Selenium tutorial, let us now understand what is parameterization, the types of parameterization and how you can use it to perform automation testing with the help of example code.

What is Parameterization in TestNG?

Role of Parameterization

In the real world, we always wish that our system works valiantly with different sets of input data. The same scenario happens when it comes to testing. Here again, we need to verify that our system is taking the set of combinations which is expected to support data-driven testing in your automated tests. Here comes the key role of 'Parameterization' in automation testing. To pass multiple information to the application at runtime, we need to parameterize our test scripts. This concept achieved by parameterization is termed as Data-driven Testing. Parameterization (Data-driven Test) is an execution approach, which lets users run a test case automatically, multiple times, with different input values. This test design will let them read data from storage, for example, from a file or database, rather than using hard-coded values. Since the Selenium WebDriver automation testing tool doesn't have any inbuilt structure or technique to parameterize the test case, we would use the TestNG testing framework.



Types of Parameterization in TestNG

To make parameterization clearer, we will go through the parameterization options using the most popular testing framework, TestNG.

There are two ways by which we can achieve parameterization in TestNG:

1. With the assistance of the Parameter annotation and the TestNG XML file
2. With the help of the DataProvider annotation

    Parameters annotation with Testng.xml file can be provided at the suite or the test level

Let's study about parameterization using TestNG and define parameters at suite and test levels in the Testng.xml file.

To do so, you need to:

- Create an XML file that will store the parameters. In the testng.xml file, we have two attributes for parameter tag, the name attribute which defines the **name** of the parameter, and the **value** attribute defines the value of the parameter.
- In the test, add annotation @Parameters

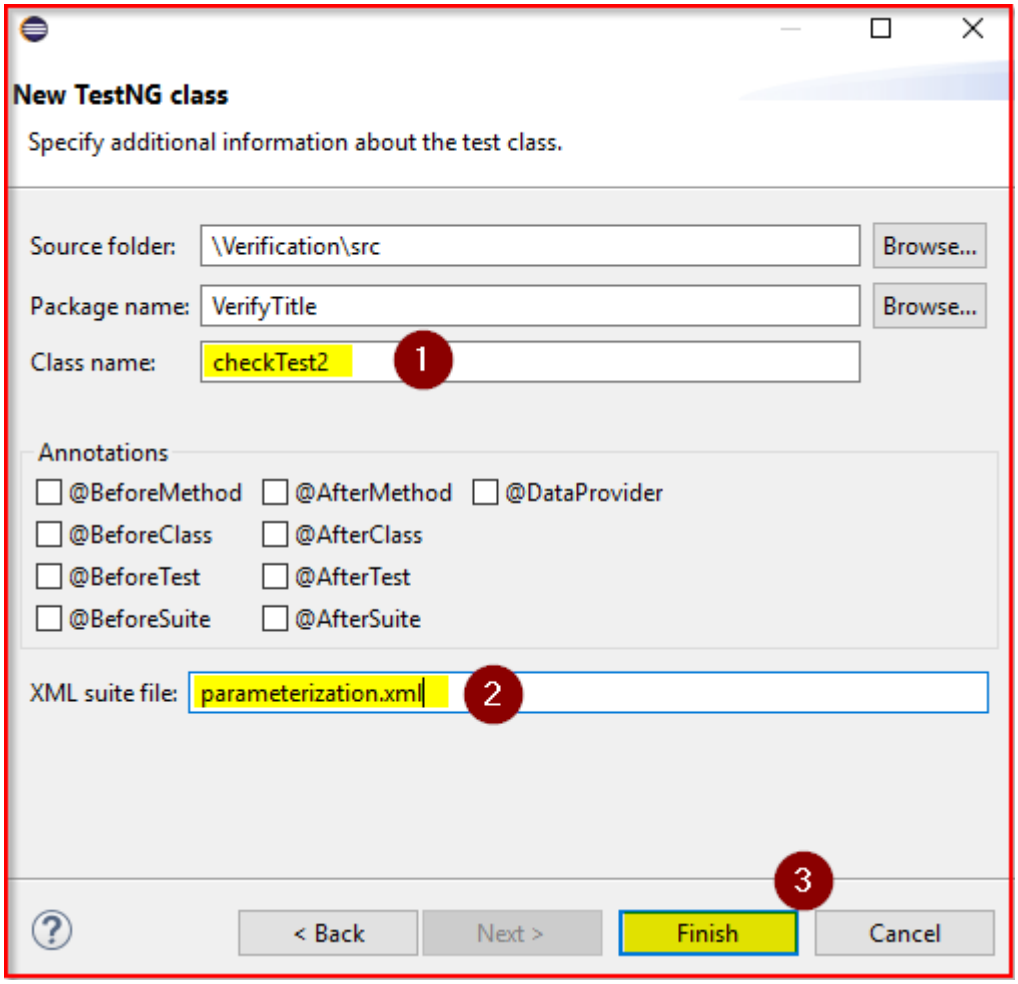
I hope, by now it is clear how to create a TestNG class with no XML file; if not, then follow these basic steps: Right-click on your **package name** > **New** > **Other** > **TestNG** > **Next** > **Finish**

Interested in learning TestNG? Check out the Selenium training in Bangalore!

Now, to create an XML file, follow these steps:

Step 1:

1. Type the **Class name** as **checkTest2**
2. Type the **XML suite file** as **xml**
3. Click on **Finish**



Step 2: Add two-parameter tags with name and value attributes, each assigned with some values to them.

Step 3: Copy and paste the below code in your local system's Eclipse IDE:

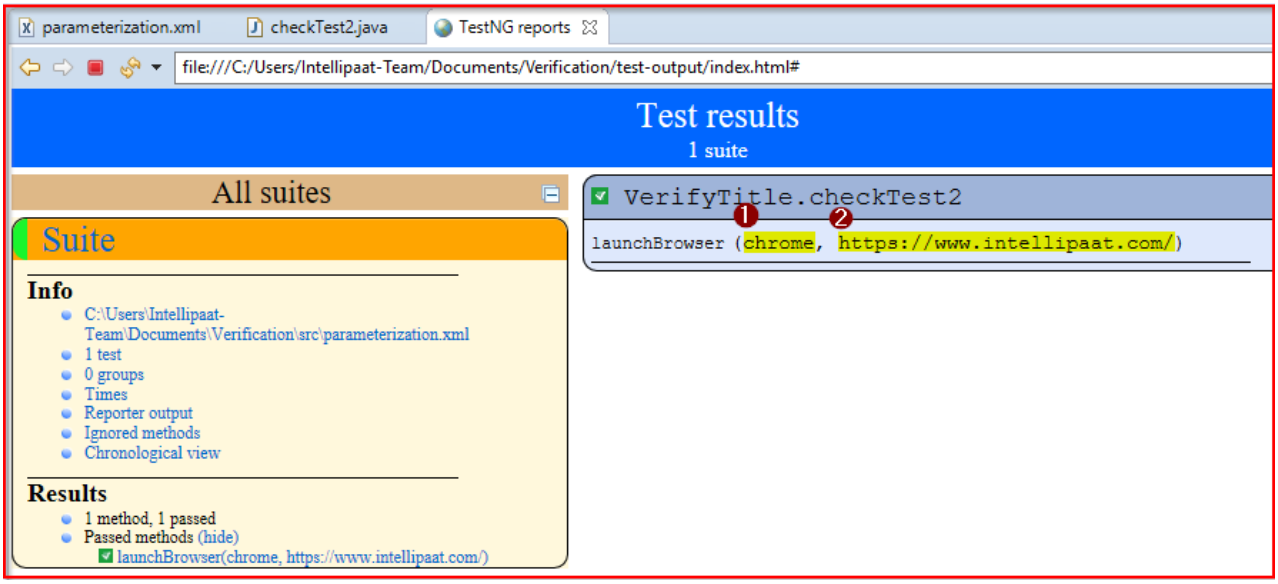
```
package VerifyTitle;


import org.testng.annotations.Parameters;

import org.testng.annotations.Test;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.ie.InternetExplorerDriver;


public class checkTest2 {

WebDriver driver;

@Parameters({"browse","URL"})

@Test

public void launchBrowser(String browse,String URL) {

switch(browse) {


case "chrome":

System.setProperty("webdriver.chrome.driver","C:\\Users\\Intellipaat-
Team\\Downloads\\driver\\chromedriver.exe");

driver = new ChromeDriver();

break;


case "firefox":

System.setProperty("webdriver.gecko.driver","C:\\Users\\Intellipaat-
Team\\Downloads\\driver\\geckodriver.exe");
```

```
driver = new FirefoxDriver();

break;


case "IE":

System.setProperty("webdriver.ie.driver","C:\\Users\\Intellipaat-
Team\\Downloads\\driver\\iexploredriver.exe");

driver = new InternetExplorerDriver();

break;

}

driver.get(URL);

System.out.println(URL);

}}
```

Get familiar with the top Selenium Interview Questions to get a head start in your career!

Step 4: Run the code from the **parameterization.xml** file, and the **index.html** test report will look like the image shown below:

1. It opens the Chrome browser
2. It navigates to 'https://www.intellipaat.com/'; the value provided is the one present in one of the parameter tags in the parameterization.xml file



In this TestNG in Selenium tutorial, so far, we learned about why TestNG came into existence, and the answer is that since Selenium framework did not have its built-in testing framework, we require the help of some external testing framework which will help us generate test reports and simplify all our testing requirements such as functional testing, regression, end-to-end testing, and more. Then we learned what TestNG is all about and its advantages over JUnit. Also, we have seen how the installation of TestNG is done and how to write our first test case. Later, we studied various TestNG annotations, viewed how to get the HTML test report, saw how to group multiple test cases, and understood what is parameterization.

If you are interested to learn Selenium on a much deeper level and want to become a professional in the testing domain, check out Intellipaat's Selenium training certification course!

Selenium CHEAT SHEET

Driver Initialization Basics

- Firefox — WebDriver driver = new FirefoxDriver();
- Chrome — WebDriver driver = new ChromeDriver();
- Internet Explorer — WebDriver driver = new SafariDriver();
- Safari Driver — WebDriver driver = new InternetExplorerDriver();

Driver Initialization Advanced

- Load Firefox from different location:

```
System.setProperty("webdriver.firfox.bin","path/to/firfox/binary");
FirefoxProfile fp= new FirefoxProfile();
```

- Load Firefox addons:

```
File file=new File("path/to/extension.xpi");
fp.addextension(file)
```

Selenium Locators

- Locating by ID

```
driver.findElement(By.id("q")).sendKeys("Se lenium 3");
```

- Locating by Name

```
driver.findElement(By.name("q")).sendKeys ("Selenium 3");
```

- Locating by Xpath

```
driver.findElement(By.xpath("//input[@id='q']")).sendKeys("Selen
ium 3");
```

- Locating Hyperlinks by Link Text

```
driver.FindElement(By.LinkText("edit this page")).Click();
```

- Locating by DOM

```
dom =document.getElementById('signinForm')
```

- Locating by CSS

```
driver.FindElement(By.CssSelector("#rightb ar >.menu>li:nth-of-type(2)>
h4"));
```

- Locating by ClassName

```
driver.findElement(By.className("profileheader"));
```

- Locating by TagName

```
driver.findElement(By.tagName("select")).C lick();
```

- Locating by LinkText

```
driver.findElement(By.linkText("NextP age")).click();
```

- Locating by PartialLinkText

```
driver.findElement(By.partialLinkText(" NextP")).click();
```

Selenium Navigators

- Navigate to url

```
driver.get("http://newexample.com")
driver.navigate().to("http://newexample.com")
```

- Refresh page

```
driver.navigate().refresh()
```

- Navigate forwards in browser history

```
driver.navigate().forward()
```

- Navigate backwards in browser history

```
driver.navigate().back()
```

TestNG

@BeforeSuite @AfterSuite @BeforeTest @AfterTest @BeforeGroups @AfterGroups @BeforeClass @AfterClass @BeforeMethod @AfterMethod

JUNIT

@After @AfterClass @Before @BeforeClass @Ignore @Test

Windows

```
String handle=driver.getWindowHandle();
Set<String> handles = getWindowHandles();
driver.switchTo().window(handle);
```

- How to switch to newly created window

```
String curWindow=driver.getWindowHandle();
```

- Get all window handles

```
Set<String> handles = getWindowHandles();
for(string handle: handles)
{
        if (!handle.equals(curWindow))
        {
        driver.switchTo().window(handle);
        }
}
```

Frames

- Using Frame Index: driver.switchTO().frame(1);
- Using Name of Frame: driver.switchTo().frame("name")
- Using web Element Object: driver.switchTO().frame(element);
- Get back to main document: driver.switchTO().defaultContent();

Operations

- Launch Webpage — driver.get("www.webdriverinselenium.com");
- Click Button — driver.findElement(By.id("submit")).click();
- Handle Alert — Alert Alertpopup = driver.switchTo().alert();
- Disable a Field — driver.getElementsByName('') [0].setAttribute('disabled','');
- Enable a Field — driver.getElementsByName('') [0].removeAttribute('disabled'; Screenshot.File snapshot=((TakesScreenshot)driver) .getScreenshotAs(OutputTypeFILE); FileUtils.copyFile(snapshot, newFile("C:\\screenshot.jpg"));
- Print Title of Page — String pagetitle = driver.getTitle(); System.out.print(pagetitle);
- Implicit Wait — driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS); Explicit Wait WebDriverWait wait=new WebDriverWait(driver, 20);
- Sleep — Thread.Sleep(10);

Alerts

```
driver.switchTO().alert.getText();
driver.switchTO().alert.accept();
driver.switchTO().alert.dismiss();
driver.switchTO().alert.sendKeys("Text");
```

Selenium Grid

- Start hub: java-jar selenium-server- standalone-x.xx.x.jar-role hub
- Start node: java-jar selenium-server- standalone-x.xx.x.jar-role node-hub
- Server: http://localhost:4444/grid/console

IntelliPaat

FURTHERMORE: Selenium Testing Training