TEST AUTOMATION - SELENIUM with JAVA

Preface:

My motivation for writing this book stems from my hands-on experience in the IT and testing domain and the experience. I have gained as an automation consultant working in numerous complex automation projects.

As part of the scope of this book we will cover Selenium WebDriver (Selenium 4.0) with Java as a programming language. We will be using Eclipse as the main IDE for creating Selenium WebDriver tests. As part of this book, we will be covering Basics of Java which would be required to use Selenium WebDriver for beginner users. As part of reporting frameworks, the book will show how to configure and use both custom JUnit and TestNG reports. My intent in this book is to discuss the key features of Selenium WebDriver. The book focuses on using Selenium WebDriver with Java language.

This book has been designed with the objective of simplicity and ease of understanding. A major fear amongst functional testers who want to learn Selenium is the fear of programming language and coding. As a part of this we will cover just enough basics on Java programming language that will give the readers confidence to use Selenium WebDriver. This book follows a unique training-based approach instead of a regular text book approach. Using a step-by-step approach, it guides the students through the exercises using pictorial snapshots. Selenium being an open-source tool needs quite a few independent components to be installed like Eclipse, TestNG, Cucumber, etc. This would usually scare testers. In this book we will cover step by step installation and configuration of each of these components.

Another differentiator is that I have tried to include many practical examples and issues which most of the automation testers encounter in day-to-day automation. These experiences will give you an insight into what challenges you could face with automation in the real world. Practical examples cover how to use most of the features within Selenium WebDriver.

The book also covers the most common interview questions on Selenium WebDriver and automation. The complete source code used in the book can be found at the following link: **https://github.com/QAKRISHNAN**

About the Author: KRISHNA N

KRISHNA is a recognized test automation engineer and corporate trainer, specializing in test automation, performance testing, security testing and test management. As an experienced corporate trainer, he has trained professionals in Selenium and other test tools across a wide range of global clients. He has designed several high-end automation frameworks including using Selenium and its integrations with tools like TestNG, JUnit, Selenium Grid, and Jenkins. He holds postgraduate degrees in MCA (Master of Computer Applications) and MTech from the Department of Computer Science.

Courses Offered: Python Full-Stack, Java Full-Stack, Software Testing [Manual & automation], Data Analytics, Cyber Security, AWS DevOps and Azure DevOps.

Contact Us:

N. Krishna 13+ Exp, MCA & MTech(CSE)

Email: n85.krishna@gmail.com

Mobile: 9493217167

****

Selenium Course Contents

Module 1: Introduction to Selenium Components

+**Automation Testing Overview & Process**

-What is automation testing

-Advantages of Automation Testing

-Limitations of Automation Testing

-Difference between Manual & Automation Testing

-Why Automation Testing is important

-When to start Automation Testing

-How to start Automation [Automation Testing Process]

-Types of Automation tools (Licensed Tools vs Open-Source Tools)

-Difference between Selenium over QTP

+**Introduction to Selenium**

-Selenium Features

-Selenium Components

-Selenium IDE and Selenium RC and WebDriver and Selenium Grid

-Difference between Selenium IDE, Selenium RC and Selenium WebDriver

Module 2: Core Java

+**Core Java Introduction**

-Introduction (History) to Java

-Types of Java Applications

-Java Platforms or Editions

-Java Version History

-Features of Java

-Differences between C and C++ and Java

-Java Software Configuration (Installation)

-Path Setting: [Java Environment Setup on Windows]

-First Java Program using NotePad

-Java Program execution using CMD

-Class and Objects

-JDK, JRE and JVM

-Download and Installation of Eclipse

-Java Project in Eclipse

-Type Casting / Type Conversion

-Separators in Java

-Operators in Java

+Control Flow Statements in Java:

-Decision-Making (Branching or Conditional) Statements

-- [if(), if-else(), else-if(), nested-if(),switch()]

-Looping (Iterative or Repetition) Statements

-- [for(), while(), do-while(), nested loops]

-Jumping Statements

-- [break, continue, return]

+Non-Primitive Data Types:

-Arrays in Java

-Strings in Java

-Java Package in Eclipse -Java Class in Eclipse -Create First Java Program in Eclipse -Scanner Class **+Java Tokens:** -Java Keywords -Java Identifiers -Constants [Literals] in Java -Literals (Constants) in Java -Variables -Data Types in Java -Comments in Java -Errors in Java	**+OOPS Concepts in Java:** -Methods in Java -Constructors in Java -Inheritance in Java -Method OverLoading in Java -Method OverRiding in Java -Polymorphism in Java -Abstraction in Java -Interface in Java -Encapsulation in Java -Garbage Collection in Java -Exception Handling in Java
Module 3: WebDriver Introduction	
- **Introduction to WebDriver** -Architecture of Selenium Webdriver -Installing Selenium WebDriver -Creating your First Script in Webdriver -Launching Chrome, Microsoft Edge and Mozilla FireFox Browsers -Difference Between WebDriver, ChromeDriver, EdgeDriver, and GeckoDrive in Selenium -Headless Browser Testing with Selenium -Verify Page Title in Selenium [Title Verification] -WebElements in Selenium -Types of Web elements in Selenium -Locaters In Selenium [Locating Strategies]	-Scroll bar [Scroll a Page] -Screenshot in Selenium -Frames in Selenium -Browser Navigations in Selenium -File Upload & Download in Selenium -Handle Multiple Windows in Selenium -Handle Alerts and Popups in Selenium -Auto Suggestions and Auto Complete in Selenium -Web Tables in Selenium -Implicit and Explicit Wait in Selenium -Actions and Action Class in Selenium -Keyboard Events in Selenium -Mouse Events in Selenium -Windows Handlers - Assertions in Selenium

+XPath -Creating and Customize XPath -Absolute Xpath, Relative Xpath and Dynamic Xpath **+Selenium Methods** -Selenium clear () Method -JavaScriptExecutor	**+AutoIT in Selenium** -How to Use AutoIT with Selenium Webdriver -Upload file in Selenium using AutoIt -Using FinderTool -AutoIT commands -Creating Scripts in AutoIT -Creating executable files -Running AutoIT Scripts from Selenium

Module 4: Framework Implementation

+TestNG Frame Work: -JUNIT and NUNIT Introduction -Features of TestNG -TestNG Installation in Eclipse -Sample TestNG Project Creation -TestNG Annotations -Executing The TestNG Script -HTML Reports -Setting Priority in TestNG -Test Suite **+Keyword Driven Framework:** -Components Of Keyword Driven Testing Framework -Framework Workflow -Project Structure -Framework Implementation -Advantages of the Framework **+Data-Driven Framework** -Creating/Opening XL Files -Reading data from XL Sheet -Writing data into XLSheet	-Executing Data Driven Tests -Analysing Test Results **+Hybrid Framework** -Creating Constant Functions -Creating Application Function Library -Preparing Keywords -Design Test Case Template -Creating Object Repository For Elements -Preparing TestData -Preparing Driver Script -Executing Driver Script -Analyzing Test Results **+Cucumber BDD Framework** -Overview of BDD and Cucumber -How to install and setup Cucumber with Eclipse -Overview of Gherkin keywords -How to create Feature file -How to generate Step Definition file -How to integrate Cucumber with Selenium WebDriver

-Counting Rows and Columns in XL Sheet -Creating ExcelUtils Class -Preparing Test Data. -Preparing Data Driven Test	+JUnit Test Runner Class -Data Driven Testing in Cucumber -Configure Cucumber with Maven and Jenkins -How generate Reports in Cucumber

Module 5: POM, GIT and JENKINS

+Maven -What is Maven and Why Maven? -Installing/Configuring Maven -Creating Maven Project -Importing Maven Project into Eclipse -What is POM.xml? -Adding Dependencies to POM.xml **+Jenkins** -Advantages of Jenkins -Steps to Install Jenkins on Windows -Why we need Jenkins	**+GIT HUB** -Create GitHub Account -Configure Git & GitHub with Eclipse -GitHub Commands +Jenkin -Installing/Configuring Jenkins -Scheduling Test Execution in Jenkins -Auto mail configuration in Jenkins -What is continues integration? -Continues integration with Jenkins

Module 6: Real Time Interview Questions and Answers

-Real Time Selenium Interview Questions & Answers

-Cucumber Framework Interview Questions & Answers

-GIT and GITHUB Interview Questions & Answers

-CI/CD and Jenkins Interview Questions & Answers

Module 7: Project Implementation

-End To End Testing on E-commerce Applications.

-End To End Testing on Banking Applications.

-End To End Testing on Social Media Applications.

-End To End Testing on Ticket Booking Applications.

-End To End Testing on Education Applications.

-End To End Testing on Online Meeting Applications.

****

AUTOMATION TESTING

SELENIUM with JAVA

Module 1: Introduction to Selenium Components

Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite. Automation Testing or Test Automation is a process of automating the manual testing process of an application or a system by using testing tools that allow you to create scripts that can be executed repeatedly, generating detailed test reports of the application or system under test. Test Automation is the best way to increase the effectiveness, test coverage, and execution speed in software testing. The primary goal of automation testing is to increase the efficiency and effectiveness of software testing by automating repetitive and time-consuming tasks.

Automation Testing is a software testing technique that performs using special automated testing software tools to execute a test case suite. Automation Testing or Test Automation is a process of automating the manual testing process of an application or a system by using testing tools that allow you to create scripts that can be executed repeatedly, generating detailed test reports of the application or system under test. Test Automation is the best way to increase the effectiveness, test coverage, and execution speed in software testing. The primary goal of automation testing is to increase the efficiency and effectiveness of software testing by automating repetitive and time-consuming tasks.

"Automation testing refers to the automatic testing of the software in which developer or tester write the test script once with the help of testing tools and framework and run it on the software. The test script automatically tests the software without human intervention and shows the result (either error, bugs are present or software is free from them)."

Automation testing is particularly useful for regression testing, which involves retesting an application after changes have been made to ensure that the changes did not introduce any new defects. By automating these tests, testers can quickly and easily verify that the application is still functioning correctly after changes have been made, without having to manually test each feature. Test Automation focuses on replacing manual human activity

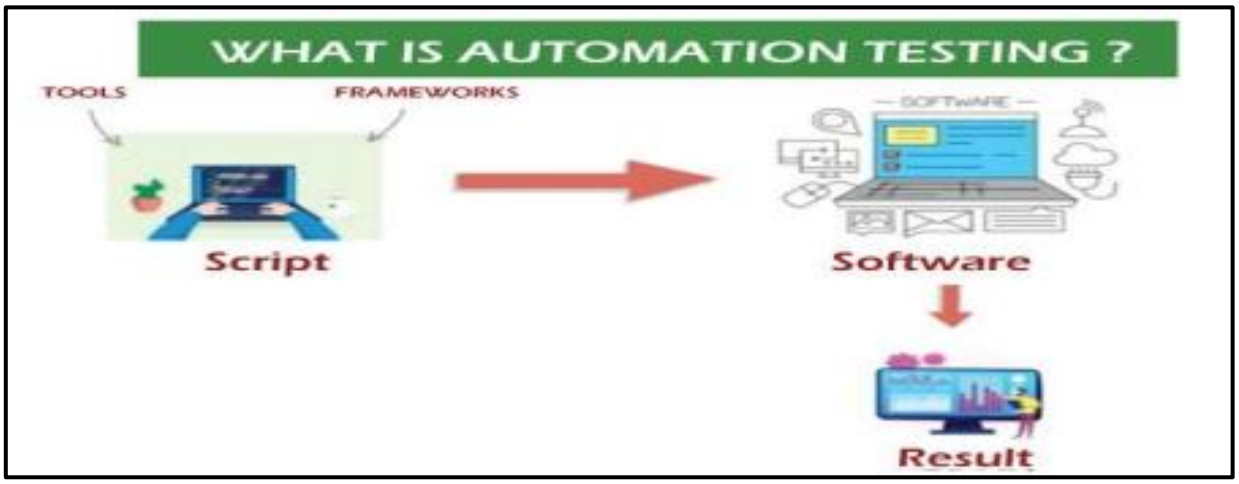
with systems or devices. Automated testing is most preferable for large projects that require testing the same areas over and over. Also, projects that have already been through an initial manual testing process.

Automation testing can be run at any time of the day. It uses scripted sequences to examine the software. It then reports on what's been found, and this information can be compared with earlier test runs. Automation developers generally write in the following programming languages: C#, Java, and Python. Test automation has many benefits for app testing cycles. This allows you to build better apps with less effort. Also, it is less time consuming. Many companies still run only manual tests as they are now aware of how to properly integrate automated testing in their app development process.

Automation testing covers both functional and performance test on an application.

• Functional automation is used for automation of functional test cases. For example, regression tests, which are repetitive in nature, are automated.

• Performance automation is used for automation of non-functional performance test cases. For example, measuring the response time of the application under considerable (say 100 users) load.



Automation Testing tools which are used for functional automation:

▪ Quick Test Professional (QTP), provided by HP.

▪ Rational Robot, provided by IBM.

▪ Coded UI, provided by Microsoft.

▪ Selenium, open source. ▪ Auto It, open Source.

Automation Testing tools which are used for non-functional automation:

▪ Load Runner, provided by HP.

▪ JMeter, provided by Apache.

▪ Burp Suite, provided by PortSwigger.

▪ Acunetix, provided by Acunetix.

Automation Testing Fundamentals:

Advantages (Benefits) of Automation Testing:

1. Automation testing offers several advantages over manual testing, including:
   Increased Efficiency: Automation testing can perform repetitive and tedious tests much faster than human testers. This increased efficiency saves time and resources, and allows testers to focus on more complex testing scenarios.
2. **Improved Test Coverage:** Automation testing can execute a large number of tests in a short amount of time, providing a higher level of test coverage than manual testing. This means that testers can test more functionality and edge cases, leading to more comprehensive testing.
3. **Saves Time:** Automating the testing process helps the testing team to use less time to validate newly created features. For instance, in manual testing, there is a need to write thousand test cases for a calculator application, but automation makes the process much faster.
4. **Less Human Resources:** Automation testing requires fewer people to perform a tedious manual test. To implement the automation test script, we need a test automation engineer who can write the test scripts to automate our tests.
5. **Running Tests 24/7:** In automation testing, we can start the testing process from anywhere in the world and anytime we want. It can also be done remotely if we don't have many approaches or the option to purchase them.
6. **Test Suite Reusability:** We can reuse the test scripts in automation testing, and we don't need to write the new test scripts again and again. These test cases can be used in various ways, as they are reusable. Reusability helps to reduce the cost and also eliminate the chances of human error.
7. **Ability to test on various platforms:** Automation testing allows the user to test the application on different web browsers and operating systems.
8. **Consistency and Accuracy:** Automation testing is not prone to human errors, ensuring consistent and accurate test results. This leads to a higher level of confidence in the quality of the software being tested.
9. **Cost-Effective:** Although the initial investment in automation testing may be high, it ultimately saves time and money in the long run. Automated tests can be re-used and executed multiple times, reducing the need for manual testing and saving on resources.

10. **Reduce the Expenses:** Automation testing is less expensive, as once the test scripts have been built, we can reuse them at any time without any extra cost. While manual testing is more expensive than automation, with manual monitoring, it is typical to execute experiments repeatedly.
11. **Fast development and Delivery:** Automated tests can be executed repeatedly and completed rapidly. We do not have to wait for weeks to execute the tests; few hours are enough for execution. Switching from manual to automation reduces the waiting time and boosts development.
12. **Early Detection of Defects:** Automation testing can be integrated into the software development process, allowing for earlier detection of defects. This can prevent issues from being introduced into later stages of development, when they are more expensive and time-consuming to fix.
13. **Productivity Improvement:** As during execution, automation tests do not require human intervention, so testing an application can be done late at night, and we can get the results next morning. Software developers and testers require less time on automation testing.
14. **Accuracy Improvement:** In manual testing, there is a chance of mistakes whether you are an experienced testing engineer. The chances of errors may increase when testing a complex use case. But Automation testing reduces the chances of errors. There is good accuracy, as we will get the same result each time on performing the same test cases.
15. **Early Bug Detection:** By automation testing, it is easy to detect critical bugs in the initial phases of software development. It reduces the cost and helps us to spend fewer working hours to fix such problems. It increases the efficiency of the team.

Overall, automation testing is a valuable tool for ensuring the quality of software applications, and can provide numerous benefits to software development teams.

Some of the other benefits of automation testing are listed as follows

- In comparison to manual testing, automation testing requires fewer resources.
- It supports the performance, functional testing, load testing, stress testing of an application or system.
- Automation testing allows the execution of test cases in a 24x7 environment.
- It facilitates the execution of repeated test cases.

- It allows the parallel execution of the test cases.
- Automation testing is 70% faster than manual testing.
- It is reliable because it tests the application with the help of tools and test scripts.
- It saves valuable time and money for the testing team involved in the project.

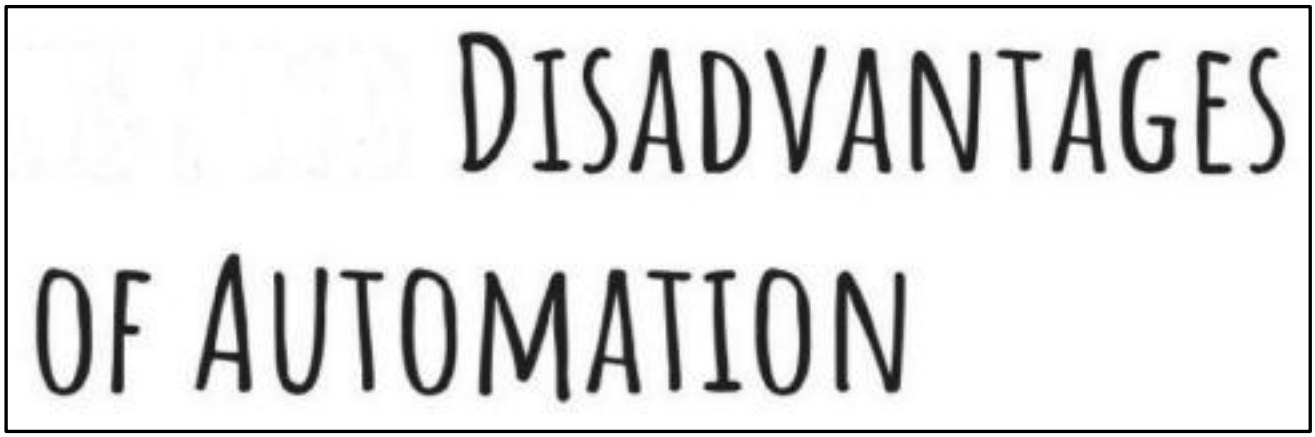
Dis-Advantages (Limitations) of Automation Testing:

- High Initial Investment: The initial cost of automation testing can be high, as it requires specialized tools, hardware, and skilled professionals. This may make it difficult for small organizations or projects to implement automation testing. One of the biggest disadvantages of automated testing is that it requires a lot of time and money to install.
- Limited Usability: Automation testing is not suitable for all types of testing. For example, it may not be effective for testing the user interface, usability, or subjective aspects of an application.
- Lack of Human Judgment: Automation testing relies on pre-defined scripts and test cases, which may not be able to replicate the judgment and intuition of a human tester. This may lead to certain defects being missed or not tested adequately.
- All types of Testing not possible (Ex: Usability, graphics, sound files): While automated tests may be used to verify most of your application's functionality, they are unsuitable for testing graphics or sound files and not possible to automate tests that verify User friendliness of the System (AUT). This is because computerized examinations often require textual descriptions to validate the result.
- Complex Applications: Automated testing may not be suitable for very complex applications, such as those with multiple layers of functionality or those that require manual testing of certain aspects.
- 100% Test automation is impractical: Generally, we try to automate maximum Test cases, not all Test cases, for some Test human user observation is required. Due to some Environment limitations, we can't automate all testable requirements.
- Tools may have their own defects: Test tool also a Software, it may have its own defects in it, so that we may not achieve desired benefits.
- Programming Knowledge is required: Every Test Tool uses any one of the Programming languages (Example UFT supports VBScript, Selenium supports Java, C# and Python) to write Test scripts. So, in order to create and edit Test Scripts Programming knowledge is mandatory. In Manual Testing, no programming knowledge is required.
- Test Tools have Environment Limitations: Test Tools have some compatibility issues with Operating Systems and browsers etc…

Example:

- UFT / QTP Supports Windows operating environment only, doesn't support other operating environments like UNIX, Macintosh etc…
- Selenium Supports Web Application test automation only, doesn't support Desktop / windows-based applications.

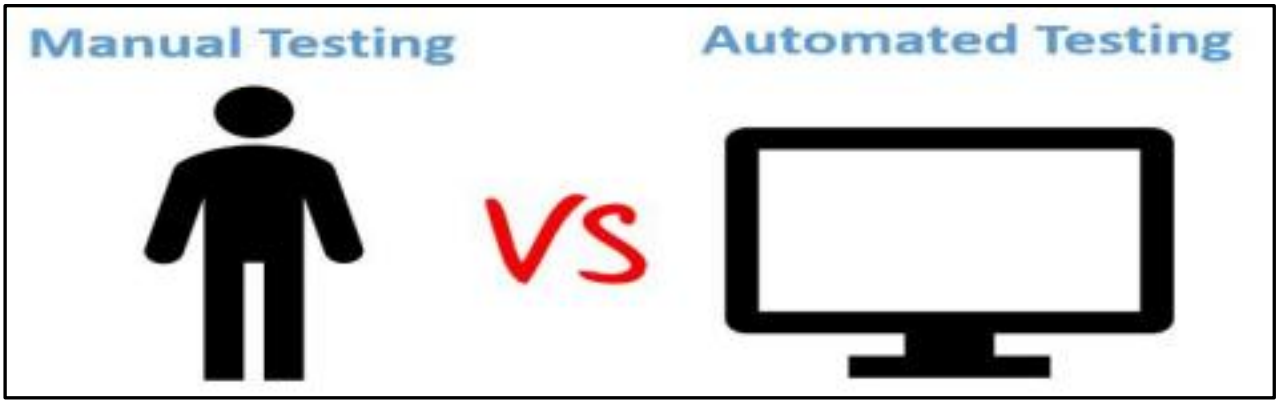
- For Manual Testing no environment limitations, we can test computer software or mobile software on any operating system and any Browser.



Difference between Manual & Automation Testing:

Automation testing	Manual testing
Testing conducted by using automated testing tools and software	Testing conducted manually by human testers
It is reliable because it tests the application with the help of tools and test scripts.	It is not reliable because there is a possibility of human error, which may not be delivered the bug-free application.
The script can be reused across multiple releases.	It could be possible when the test case only needs to run once or twice.
The execution is always faster than the manual; that's why the automation testing process is time-saving.	It is time consuming due to the usage of the human resources.
Without having an understanding of programming language, we cannot write the test script.	There is no need-to-know programming language but should have the product knowledge to write the test case.
Automation testing cannot give the guarantee of user-friendliness.	Manual testing checks the user-friendliness.
Automation testing uses frameworks like Data Drive, Keyword etc.	Manual testing doesn’t use frameworks.
Automation does not allow random testing.	Exploratory testing is possible in Manual Testing.

Done by tools. Its accurate and never gets bored!	Repetitive Manual Test Execution can get boring and error-prone.
Testers set up frameworks and create test scripts that automate user actions required for testing a website or app.	Testers manually scour through different features of a website or app in order to identify bugs, errors, anomalies, and the like.



Why Automation Testing is important:

- **Running Tests 24/7:** You can start the test from anywhere in the world and anytime you want to. You can even do that remotely if you don't have a lot of devices or you don't have the possibility to buy them.
- **Fewer Human Resources:** You just need a test automation engineer to write your scripts to automate your tests, instead of a lot of people doing boring manual tests over and over again.
- **Reusability:** The scripts are reusable and you don't need new scripts every time. Also, you can redo the steps that are exactly as the previous ones.
- **Bugs:** Automation helps you find bugs in the early stages of software development, reducing expenses and working hours to fix these problems as well.
- **Reliability:** Automated testing is more reliable and way quicker when running boring repetitive standardized tests which cannot be skipped, but may cause errors when manually tested.
  - The application has a very vast area with a high degree of investing effort in regression.
  - Optimization in costs occurred due to manual errors.
  - The software has multiple versions and releases.
  - It is cost effective in long run.
  - The risk factor is higher for a broader scope of test execution.
  - Cost figures and mathematical calculations are included in the software functionality.
  - There is a greater increase in the execution tempo, efficiency along with the software quality.
  - There is a lesser turnaround time, even for high-risk software testing.

When to start Automation Testing:

a. **Test cases are stable and well-defined:** Automation testing requires well-defined test cases that have been fully tested and validated manually. These test cases should be stable and not subject to frequent changes.
b. **Test cases are repetitive:** Automation testing is most effective when used for repetitive tasks, such as regression testing, load testing, and performance testing.
c. **Budget and resources are available:** Automation testing requires investment in tools, infrastructure, and skilled resources. Therefore, it is important to ensure that budget and resources are available to.
d. **Project timeline allows for automation:** Automation testing requires time and effort to set up, implement, and maintain. Therefore, it is important to ensure that the project timeline allows for automation testing to be properly executed.
e. **Business value can be achieved:** Automation testing should be implemented only if it can provide business value by improving testing efficiency, reducing costs, increasing test coverage, and improving overall software quality.

In summary, automation testing should be started once manual test cases are stable and well-defined, repetitive, budget and resources are available, project timeline allows for automation, and business value can be achieved.

How to start Automation [Automation Testing Process]:

Even if all the above situations make sense to go for Automation Testing, the fact is you'll need resources to process it successfully. The automation testing process is a systematic approach to organize and execute testing activities in a manner that provides maximum test coverage with limited resources. There are important steps on which an Expert test team should pay attention in an Automation Process:

1. **Selection of Automation Test Tool:** There are various tools in the market, each with its own strengths and weaknesses, but you have to select those which are best for your application. It mainly depends on the technology of that Application Under Test (AUT) to which you are planning to Automate. It'll great if you do conduct Proof of Concept of Tool first, thus you can ensure that test coverage can be maximize.

   Some popular automation tools are:

- **QTP:** HP's Quick Test Professional – functional testing tool which support for both Web & Desktop Applications.
- **Selenium:** An open-source automated testing suite for web applications supports mostly all browsers.

2. **Training the Team:** After tool selection and resource hiring, the next step is logically the training of the resources. If manual testers are converted into automation engineers, they have to be trained on automation terminologies and concepts. If an automation architect is hired from outside, he must get knowledge about the product to test, the manual testing process, and what management is expecting. Train them on the tools which the organization is already using bug tracking software and requirements

management software. Good training and strong communication between manual testers, developers, and the automation team is really necessary.

3. **Plan Automation Testing:** Plan automation testing by identifying the test cases that will be automated, creating test scripts, design of Framework, setting up the test environment, and establishing the criteria for success.
4. **Set up the automation testing environment:** Set up the automation testing environment by installing and configuring the automation testing tool, integrating it with the development environment, and setting up the necessary hardware and software infrastructure.
5. **Create and execute automated test scripts:** Create and execute automated test scripts to test the application or system under test. Test scripts should cover all the test cases identified in the planning phase.
6. **Maintain and update the automation testing environment:** Maintain and update the automation testing environment to ensure that it remains up-to-date and effective. Update test scripts as necessary, and continuously improve the automation testing process.

Types of Automation Tools (Licensed Tools & Open-Source Tools):

There are several types of automation tools available, including both licensed and open-source tools.

1) Selenium: Selenium is a popular open-source automation tool used for functional and regression testing of web applications. It was first developed in 2004 and has since become one of the most widely used automation tools in the industry. Selenium allows users to write test scripts in multiple programming languages, including Java, Python, C#, Ruby, and JavaScript. It supports multiple web browsers, including Chrome, Firefox, Safari, and Edge, and can be used to test web applications on multiple platforms, including Windows, macOS, and Linux.

2) QTP: QTP (QuickTest Professional), now known as UFT (Unified Functional Testing), is a commercial automation testing tool developed by Micro Focus. UFT is a popular commercial automation tool used for functional and regression testing of desktop, web, and mobile applications.

3) TestComplete: TestComplete is a commercial automation tool used for functional and regression testing of desktop, web, and mobile applications. It supports multiple scripting languages, including Python, VBScript, and JavaScript.

4) Robot Framework: Robot Framework is a generic open-source automation framework used for acceptance testing and acceptance test-driven development (ATDD).

5) Appium: Appium is a popular open-source automation tool used for functional and regression testing of mobile applications. It supports multiple platforms, including Android and iOS.

6) Cucumber: Cucumber is an open-source automation tool used for acceptance testing and Behavior Driven Development (BDD). It supports multiple programming languages, including Java and Ruby.

7) JMeter: JMeter is an open-source automation tool primarily used for load testing and performance testing of web applications, web services, and other network applications. It was originally developed for testing web applications but has since been expanded to support a wide range of testing scenarios. JMeter works by simulating a large number of concurrent users accessing a web application or service, and measuring how the application responds under different load conditions.

QTP and Selenium are the most used tools in the market for software automation testing.

Difference between Selenium over QTP:

Selenium	QTP
Selenium is an open-source tool.	QTP is a commercial tool and there is a cost involved in each one of the licenses.
Can be extended for various technologies that expose DOM.	Limited add-ons and needs add-ons for each one of the technologies.
Has capabilities to execute scripts across different browsers.	Can run tests in specific versions of Firefox, IE, and Chrome.
Can execute scripts on various operating systems.	Works only with Windows.
Executes tests within the browser, so focus is NOT required while script execution is in progress.	Needs Focus during script execution, as the tool acts on the browser (mimics user actions).
Supports mobile devices.	Supports mobile devices with the help of third-party tools.
Can execute tests in parallel with the use of Selenium Grids.	QTP cannot execute tests in parallel, however integrating QTP with QC allows testers to execute in parallel. QC is also a commercial tool.

****

Introduction to Selenium

Selenium is one of the most widely used open-source Web UI (User Interface) automation testing suite. Selenium is an open-source and a portable automated software testing tool for testing web applications across multiple browsers. It has capabilities to operate across different browsers and operating systems. Selenium is not just a single tool but a set of tools that helps testers to automate web-based applications more efficiently. It's primarily built in Java and supports several browsers and programming languages. You can use multiple programming languages like Java, C#, Python, etc to create Selenium Test Scripts. Selenium Web driver is most popular with Java and C#. Selenium test scripts can be coded in any of the supported programming languages and can be run directly in most modern web browsers. Browsers supported by Selenium include Internet Explorer, Mozilla Firefox, Google Chrome and Safari.

Before learning the concepts of Selenium, you should have a basic understanding of JAVA or any other object-Oriented programming language. In addition, you should have prior knowledge of software testing techniques like Manual testing, Automation testing, functional testing, etc.

Primarily, Selenium was created by Jason Huggins in 2004. An engineer at Thought Works, he was working on a web application that required frequent testing. Having realized that their application's repetitious Manual Testing was becoming increasingly inefficient, he created a JavaScript program that would automatically control the browser's actions. He named this program the "JavaScriptTestRunner". Seeing potential in this idea to help automate other web applications, he made JavaScript Runner open-source, which was later re-named Selenium Core.

Selenium Features:

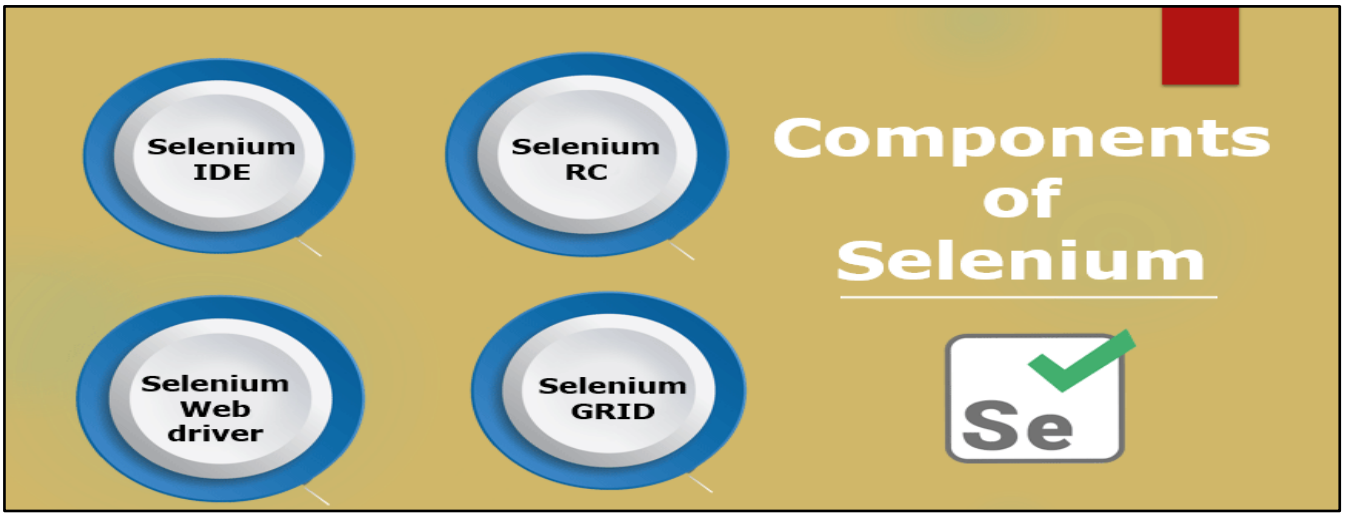
Multi-Browser Support:	Selenium is capable of interacting with web apps and the web elements in a browser just like a real user would. Selenium does that by using a browser native support that makes direct calls without the use of any intermediary device or software. Almost all browsers are supported by Selenium – Chrome, Safari, IE, Opera, Edge, and Firefox.
Multi-Language Compatibility:	Selenium supports almost all programming languages like Java, PHP, Python, JavaScript, Perl, Ruby, etc. You can write automation test scripts using any programming language you feel comfortable with. You can also use switch statements, conditional statements, or decision-making statements to enhance your automation test script. This step will make your test script capable of handling all kinds of situations.
Portability (Ability to work with	Selenium is portable software. It can work with different Operating Systems, Browsers and Programming Languages. Following is the list:

different Operating Systems):	Programming Languages: C#, Java, Python, PHP, Ruby, Perl, and JavaScript. Operating Systems: Android, iOS, Windows, Linux, Mac, Solaris. Browsers: Google Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera, Safari, etc Selenium allows users to automate test scripts across different devices like Android, iPhone, etc. Selenium can also be integrated with testing frameworks like TestNG for application testing and generating reports. Selenium can be integrated with frameworks like Ant and Maven for source code compilation.
Performance and Speed:	Selenium has a particular component for the automation of web app testing called WebDriver. This tool is able to execute test cases quicker than the other tools. It is capable of communicating directly with the browser so there is no requirement for intermediaries like the server.
Dynamic Web Elements:	Selenium is capable of handling dynamic web elements with ease. It utilizes some of the following methods to do that: Contains (): You can use a partial text to find an element. Absolute XPath (): This XPath can easily handle dynamic web elements. It comes with a complete set of paths for web UI automation, right from the root node. StartsWith(): This function helps find an attached attribute to a dynamic web element by matching or finding the starting text.
Reusability and Extras:	All the scripts written with the help of Selenium are capable of supporting browser compatibility testing. The extra plugins help widen the scope of application testing and they can be customized.
Take less time to execute a test:	Selenium reduces the test execution time. This helps make the execution more reliable and faster. It also supports parallel test execution which reduces time and increases the efficiency of tests. Selenium requires fewer resources as compared to other automation test tools.
Server installation is not required:	You don't need to install a server for Selenium. Selenium can interact directly with the browser.
Selenium WebDrivers:	Selenium WebDriver is an important tool offered by Selenium. It provides a lot of solutions for some potential problems in automation testing. It also helps testers deal with complex web elements like radio buttons, dropdowns, alerts, etc. by using dynamic locators.

Selenium Components [Selenium Tool Suite]:

Selenium is a powerful tool for controlling web browser through program. It is functional for all browsers, works on all major OS and its scripts are written in various languages i.e Java, Python, C#, etc, we will be working with Python. Selenium is not just a single tool but a suite of software, each with a different approach to support automation testing.

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



1) Selenium IDE:

Selenium Integrated Development Environment (IDE) is a Firefox plugin that lets testers to record their actions as they follow the workflow that they need to test. The Selenium-IDE (Integrated Development Environment) is an easy-to-use as a Firefox Add-On that you can install as easily as you can with other plugins. It provides a Graphical User Interface for recording user actions using Firefox. Selenium Integrated Development Environment (IDE) is the simplest framework in the Selenium suite and is the easiest one to learn. It is a complete integrated development environment (IDE) for Selenium tests. It was previously known as Selenium Recorder. Selenium-IDE was originally created by Shinya Kasatani and donated to the Selenium project in 2006. Selenium IDE was previously little-maintained.

Selenium IDE began being actively maintained in 2018. However, the recorded scripts can be converted into various programming languages supported by Selenium and the scripts can be executed on other browsers as well.

In Selenium IDE, the test case execution is very slow, and the report generation step for the test cases is not good as compared with other components. It doesn't support test case execution in parallel or remote execution. If you wish to create scripts using Selenium IDE, you need to use Selenium RC or Selenium WebDriver to write more advanced and robust test cases.

Some of the features of Selenium IDE:

- The recording, debugging and editing of the functional tests can be done in Selenium IDE.
- The scripts in Selenium IDE are developed in Selenese which is a scripting language in Selenium.
- Selenium commands help us to perform tasks like clicking a button or link, taking input in an edit box, obtaining a text from a web element and so on.

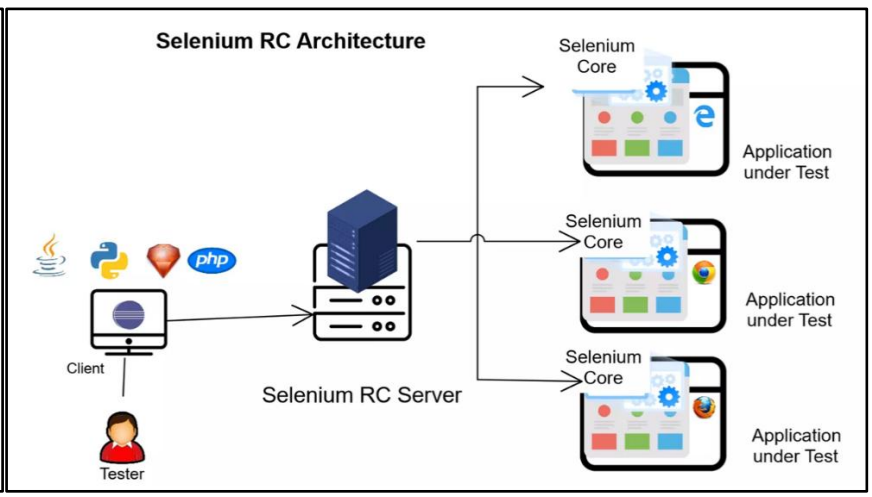
Few drawbacks of Selenium IDE are:

- It restricts the test case execution to the Firefox browser.
- It doesn't extend the support to mobile-based testing like iPhone/Android testing.
- The execution of test cases is very slow and the report generation step is not good when compared with other components.

2) Selenium RC:

Selenium Remote Control (RC) was the flagship testing framework that allowed more than simple browser actions and linear execution. It makes use of the full power of programming languages such as Java, C#, PHP, Python, Ruby and PERL to create more complex tests. This is the first automated web testing tool that allows users to use a programming language they prefer. Selenium RC, also known as Selenium 1, was the main Selenium project for a long time before the WebDriver merge brought up Selenium 2. It mainly relies on JavaScript for automation.

Selenium Remote Control (RC) was the main Selenium project that sustained for a long time before Selenium WebDriver (Selenium 2.0) came into existence. Now Selenium RC is hardly in use, as WebDriver offers more powerful features, however users can still continue to develop scripts using RC. It allows us to write automated web application UI tests with the help of full power of programming languages such as Java, C#, Perl, Python and PHP to create more complex tests such as reading and writing files, querying a database, and emailing test results. Selenium RC works in such a way that the client libraries can communicate with the Selenium RC Server passing each Selenium command for execution. Then the server passes the Selenium command to the browser using Selenium-Core JavaScript commands.



Features of Selenium RC:

- It is based on JavaScript. It doesn't support a Record/Playback feature.
- It is based on a client/server architecture.
- To initiate test execution, we have to create an instance of the Selenium RC server and you need to start the server manually.
- It supports parallel execution of test cases as well as remote execution with the help of Selenium Grid.

"The drawback of Selenium RC is that whenever you want to execute the test cases, you should start Selenium Standalone server manually. In order to overcome this problem, Selenium Webdriver was introduced".

3) Selenium WebDriver:

Selenium WebDriver is the successor to Selenium RC which sends commands directly to the browser and retrieves results. WebDriver is a tool for automating testing web applications. It is popularly known as Selenium 2.0. WebDriver uses a different underlying framework, while Selenium RC uses JavaScript Selenium-Core embedded within the browser which has got some limitations. WebDriver interacts directly with the browser without any intermediary, unlike Selenium RC that depends on a server. The WebDriver proves to be better than Selenium IDE and Selenium RC in many aspects. It implements a more modern and stable approach in automating the browser's actions. Selenium WebDriver is a browser automation framework that accepts commands and sends them to a browser. It is implemented through a browser-specific driver. It directly communicates with the browser and controls it. Selenium WebDriver supports various programming languages like – Java, C#, PHP, Python, Perl, Ruby. and Javascript. WebDriver directly calls the methods of different browsers hence we have separate driver for each browser.

Difference between Selenium IDE, Selenium RC and Selenium WebDriver:

SELENIUM IDE	SELENIUM RC	SELENIUM WEBDRIVER
It only works in Mozilla browser.	It supports with all browsers like Firefox, IE, Chrome, Safari, Opera etc.	It supports with all browsers like Firefox, IE, Chrome, Safari, Opera etc.

It supports Record and playback	It doesn't support Record and playback	It doesn't support Record and playback
Doesn't required to start server before executing the test script.	Required to start server before executing the test script.	Doesn't required to start server before executing the test script.
It is a GUI Plug-in	It is standalone java program which allow you to run Html test suites.	It is actual core API which has binding in a range of languages.
Core engine is JavaScript based	Core engine is JavaScript based	Interacts natively with browser application
Very simple to use as it is record & playback.	It is easy and small API	As compared to RC, it is bit complex and large API.
It is not object oriented	API's are less Object oriented	API's are entirely Object oriented
It doesn't support of moving mouse cursors.	It doesn't support of moving mouse cursors.	It supports of moving mouse cursors.

4) Selenium Grid:

Selenium Grid is a tool used to run parallel tests across different machines and different browsers simultaneously which results in minimized execution time. Selenium Grid is a tool used together with Selenium RC to run parallel tests across different machines and different browsers all at the same time. Parallel execution means running multiple tests at once. This implies –running multiple tests at the same time against different machines running different browsers and operating systems. This is done by routing commands to remote web browser instances, where one server acts as the hub. This hub routes test commands that are in JSON format to multiple registered Grid nodes. Hub enables simultaneous execution of tests on multiple machines, managing different browsers centrally, instead of conducting different tests for each of them. Selenium Grid makes cross browser testing easy as a single test can be carried on multiple machines and browsers, all together, making it easy to analyze and compare the results. Selenium Grid has 2 versions – the older Grid 1 and the newer Grid 2.

The two major components of the Selenium Grid architecture are:

1. Hub is a server that accepts the access requests from the WebDriver client, routing the JSON test commands to the remote drives on nodes. It takes instructions from the client and executes them remotely on the various nodes in parallel

2. Node is a remote device that consists of a native OS and a remote WebDriver. It receives requests from the hub in the form of JSON test commands and executes them using WebDriver

Testers should use Selenium Grid in the following circumstances:

To run tests on multiple browsers and their versions, different devices, and operating systems to reduce the time that a test suite takes to complete execution

Features:

- Enables simultaneous running of tests in multiple browsers and environments.
- Saves time enormously.
- Utilizes the hub-and-nodes concept. The hub acts as a central source of Selenium commands to each node connected to it.

****

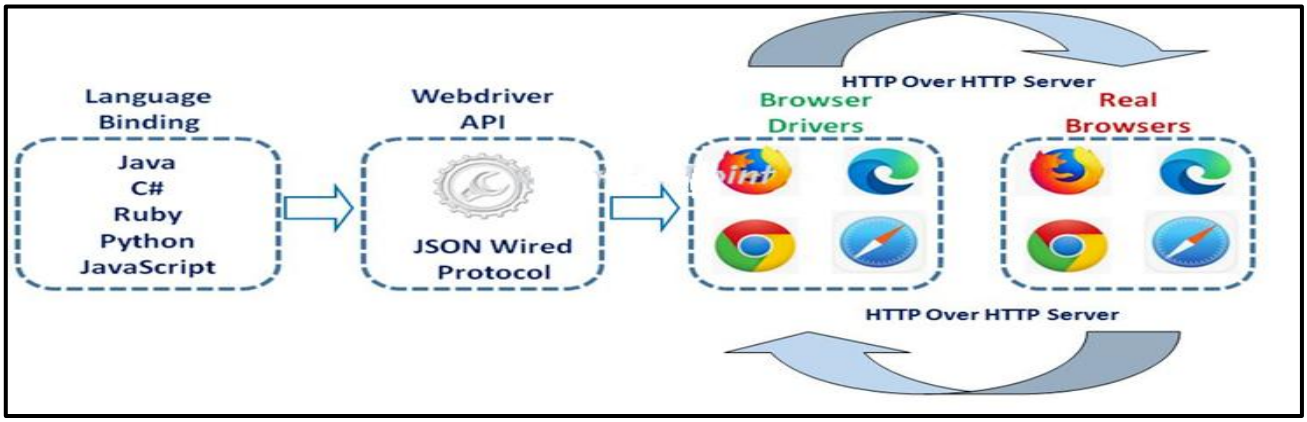
Introduction to Selenium WebDriver

Selenium is one of the most widely used open-source Web UI (User Interface) automation testing suite. Selenium is an open-source and a portable automated software testing tool for testing web applications across multiple browsers. It has capabilities to operate across different browsers and operating systems. Selenium is not just a single tool but a set of tools that helps testers to automate web-based applications more efficiently. It's primarily built in Java and supports several browsers and programming languages. You can use multiple programming languages like Java, C#, Python, etc to create Selenium Test Scripts. Selenium Web driver is most popular with Java and C#. Selenium test scripts can be coded in any of the supported programming languages and can be run directly in most modern web browsers. Browsers supported by Selenium include Internet Explorer, Mozilla Firefox, Google Chrome and Safari.

Before learning the concepts of Selenium, you should have a basic understanding of JAVA or any other object-oriented programming language. In addition, you should have prior knowledge of software testing techniques like Manual testing, Automation testing, functional testing, etc.

Primarily, Selenium was created by Jason Huggins in 2004. An engineer at Thought Works, he was working on a web application that required frequent testing. Having realized that their application's repetitious Manual Testing was becoming increasingly inefficient, he created a JavaScript program that would automatically control the browser's actions. He named this program the "JavaScriptTestRunner". Seeing potential in this idea to help automate other web applications, he made JavaScript Runner open-source, which was later re-named Selenium Core.

Selenium WebDriver is the successor to Selenium RC which sends commands directly to the browser and retrieves results. WebDriver is a tool for automating testing web applications. It is popularly known as Selenium 2.0. WebDriver uses a different underlying framework, while Selenium RC uses JavaScript Selenium-Core embedded within the browser which has got some limitations. WebDriver interacts directly with the browser without any intermediary, unlike Selenium RC that depends on a server. The WebDriver proves to be better than Selenium IDE and Selenium RC in many aspects. It implements a more modern and stable approach in automating the browser's actions. Selenium WebDriver is a browser automation framework that accepts commands and sends them to a browser. It is implemented through a browser-specific driver. It directly communicates with the browser and controls it. Selenium WebDriver supports various programming languages like – Java, C#, PHP, Python, Perl, Ruby. and Javascript. WebDriver directly calls the methods of different browsers hence we have separate driver for each browser.



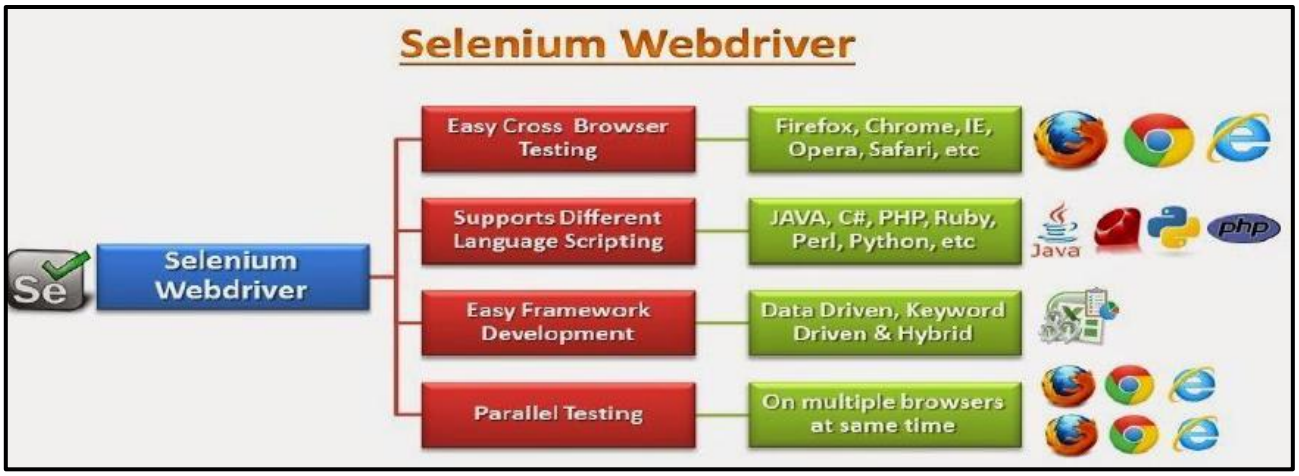
Some of the most widely used web drivers include:

- Mozilla Firefox Driver (Gecko Driver)
- Google Chrome Driver
- Internet Explorer Driver
- Opera Driver
- Safari Driver
- HTML Unit Driver (a special headless driver)

Selenium WebDriver also supports the following:

- Operation System Support – Windows, Mac OS, Linux, Solaris

WebDriver is best explained with a simple architecture diagram as shown below.



The following table lists some of the most frequently used commands in WebDriver along with their syntax:

S.No.	Command	Description
1	driver.get("URL")	To navigate to an application.
2	driver.getTitle();	To get the actual title of any web page.
3	element.sendKeys("inputtext")	Enter some text into an input box.
4	element.clear()	Clear the contents from the input box.

5	select.deselectAll()	Deselect all OPTIONs from the first SELECT on the page.
6	select.selectByVisibleText("some text")	Select the OPTION with the input specified by the user.
7	driver.switchTo().window("windowName")	Move the focus from one window to another.
8	driver.switchTo().frame("frameName")	Swing from frame to frame.
9	driver.switchTo().alert()	Helps in handling alerts.
10	driver.navigate().to("URL")	Navigate to the URL.
11	driver.navigate().forward()	To navigate forward.
12	driver.navigate().back()	To navigate back.
13	driver.close()	Closes the current browser associated with the driver.
14	driver.quit()	Quits the driver and closes all the associated window of that driver.
15	driver.refresh()	Refreshes the current page

Installing Selenium WebDriver:

WebDriver is the most important Selenium component. It is a web framework that allows the tester to execute cross-browser testing. To create the test scripts, there is an option to choose a programming language, such as Java, JavaScript, Python, and C#/Ruby. Selenium WebDriver is designed in accordance with the client/server model. It communicates with the browser through the driver and data is communicated through the HTTP protocol.

1. **Download and Install Java 8 or Higher Version.**
2. **Download and Configure Eclipse or any Java IDE of your choice.**
3. **Download Selenium WebDriver Java Client**
4. **Download Any Browser and Driver**
5. **Configure Selenium WebDriver**

1. Download and Install Java:

Step1: First of all, go to Oracle's official website to download Java.

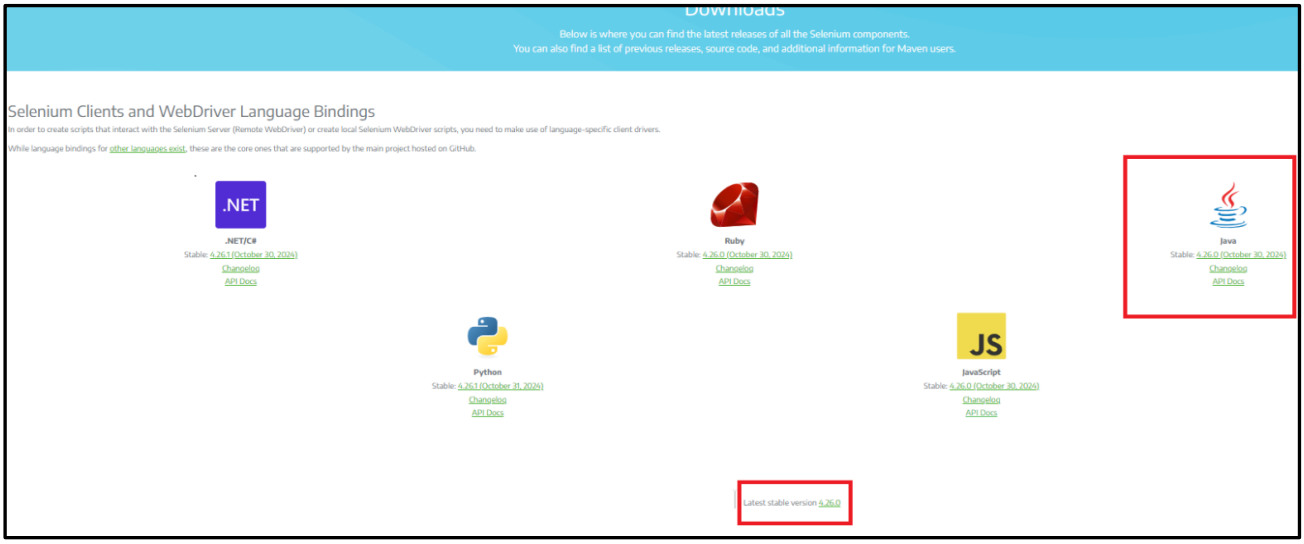
Step2: Then download it by clicking on JDK Download and then click on the one you want to download according to your system i.e., 32 or 64-bit version.
Step3: Then just execute the installer and Java will be downloaded to your system.
Step4: After installation of Java, you need to set path or configure the environment variables in your system.
Note: Detailed Java Installation and Path setting process steps given at Core Java Part [Please go through it].

2. Download and Configure Eclipse IDE:

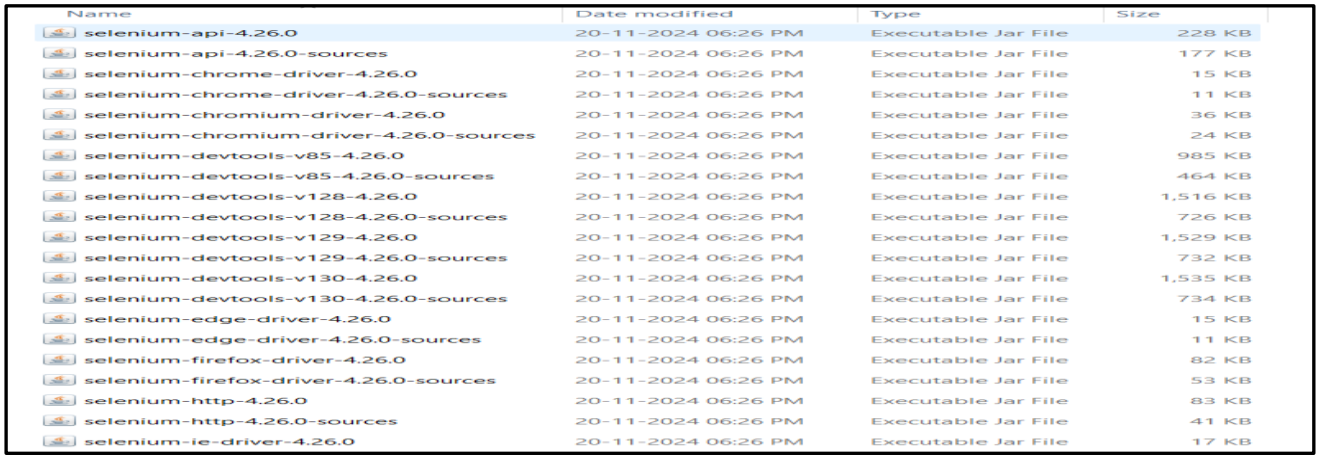
Step1: The first step, go to the official website of Eclipse and click on the download button.
Step2: It will redirect you to the "Download Packages" section. Scroll down through the webpage and click on "Eclipse IDE for Java Developers".
Step3: Now, click on the "Download x86_64" button. You can also select other options to download based on the operating system you are currently working on.
Step4: After downloading, while installing click on "Eclipse for Java Developers" and after that click on Install.
Step5: To configure the workspace, select a convenient directory where you want to keep all of your Selenium trails and click on Launch button.
Step 6: It will launch the default interface of Eclipse IDE.
Note: Detailed Eclipse IDE Installation process steps given at Core Java part [Please go through it].

Download Selenium WebDriver Java Client:

Step 1: First of all, open your browser and go to the official selenium website using this link: **https://docs.seleniumhq.org/download /**
Step 2: Now download the Selenium WebDriver Java client, click on the "Download" link of Java Client Driver as shown in the image given below.



Step 3: After downloading, the downloaded file would be in zipped format. Extract the contents in a convenient directory. It contains the essential jar files required to configure Selenium WebDriver in Eclipse IDE.



Name	Date modified	Type	Size
selenium-api-4.26.0	20-11-2024 06:26 PM	Executable Jar File	228 KB
selenium-api-4.26.0-sources	20-11-2024 06:26 PM	Executable Jar File	177 KB
selenium-chrome-driver-4.26.0	20-11-2024 06:26 PM	Executable Jar File	15 KB
selenium-chrome-driver-4.26.0-sources	20-11-2024 06:26 PM	Executable Jar File	11 KB
selenium-chromium-driver-4.26.0	20-11-2024 06:26 PM	Executable Jar File	36 KB
selenium-chromium-driver-4.26.0-sources	20-11-2024 06:26 PM	Executable Jar File	24 KB
selenium-devtools-v85-4.26.0	20-11-2024 06:26 PM	Executable Jar File	985 KB
selenium-devtools-v85-4.26.0-sources	20-11-2024 06:26 PM	Executable Jar File	464 KB
selenium-devtools-v128-4.26.0	20-11-2024 06:26 PM	Executable Jar File	1,516 KB
selenium-devtools-v128-4.26.0-sources	20-11-2024 06:26 PM	Executable Jar File	726 KB
selenium-devtools-v129-4.26.0	20-11-2024 06:26 PM	Executable Jar File	1,529 KB
selenium-devtools-v129-4.26.0-sources	20-11-2024 06:26 PM	Executable Jar File	732 KB
selenium-devtools-v130-4.26.0	20-11-2024 06:26 PM	Executable Jar File	1,535 KB
selenium-devtools-v130-4.26.0-sources	20-11-2024 06:26 PM	Executable Jar File	734 KB
selenium-edge-driver-4.26.0	20-11-2024 06:26 PM	Executable Jar File	15 KB
selenium-edge-driver-4.26.0-sources	20-11-2024 06:26 PM	Executable Jar File	11 KB
selenium-firefox-driver-4.26.0	20-11-2024 06:26 PM	Executable Jar File	82 KB
selenium-firefox-driver-4.26.0-sources	20-11-2024 06:26 PM	Executable Jar File	53 KB
selenium-http-4.26.0	20-11-2024 06:26 PM	Executable Jar File	83 KB
selenium-http-4.26.0-sources	20-11-2024 06:26 PM	Executable Jar File	41 KB
selenium-ie-driver-4.26.0	20-11-2024 06:26 PM	Executable Jar File	17 KB

Note: You can download Selenium WebDriver single file also [Not a ZIP file].

4. Download Any Browser and Driver

How To Download ChromeDriver for Selenium:

A ChromeDriver is a standalone server or a separate executable used by Selenium WebDriver to control Chrome. Running Selenium test scripts on the Google Chrome browser is impossible without ChromeDriver. ChromeDriver is a separate executable that Selenium WebDriver uses to control Chrome. One can easily initialize the object of ChromeDriver using the following command:

WebDriver driver = new ChromeDriver

We assume that you have already installed the Google Chrome browser. The next step is to find the appropriate version of the ChromeDriver. Chromedriver is a .exe file that your WebDriver interface uses to initiate the Google Chrome browser. As this is an open tool, you can download it from its official website or the Selenium community. The only point that you need to consider is that the version of your Chrome browser should be compatible with the chromedriver.exe that you are going to download.

Below are the steps to follow while configuring the chrome setup for Selenium:

#1) Check the version of the chrome.

Open Chrome Browser -> Help -> About Google Chrome

#2) Displays Google Chrome current version.

Download ChromeDriver:

Users first need to download ChromeDriver for their respective Operating systems from this link[https://developer.chrome.com/docs/chromedriver/] you will see the latest ChromeDriver for the latest google chrome version.

#3) Once the zip file is downloaded for the operating system, unzip it to retrieve the chromedriver.exe executable file. Copy this file to a specific location of your choice.

#4) The path of the chromedriver ("C:\\Selenium 2023\\chromedriver_win32 \\chromedriver.exe") will be used in our program.

Creating your First Script in Webdriver:

In this section, you will learn how to run your Selenium Test Scripts on Chrome Browser.

Chrome browser implements the WebDriver protocol using an executable called ChromeDriver.exe. This executable starts a server on your system which in turn is responsible for running your test scripts in Selenium.

Let us consider a test case in which we will try to automate the following scenarios in Google Chrome browser.

A)	Launch the Chrome browser.
B)	Maximize the browser.
C)	Open URL: https://www.google.co.in.
D)	Wait for the page to load.
E)	Print a message (URL is opened successfully) on the console.
F)	Close the browser.

Important Steps to Automate Test Script:

Step1:	To create your first Selenium WebDriver script, first, open your Eclipse IDE.
Step2:	Create a Java Project. Go to File > New > Java Project and write "MYPRACTICEAUTOMATIONPROGRAMS".
Step3:	Create a Package. Go to src > Right-click > New > Package and declare a name "programs".
Step4:	Now create a Java class. Go to package "newProject" > Right-click > New > Class and declare a class name "OpenGoogleURL".
Ste5:	Now we will create an object of a class ChromeDriver by taking WebDriver reference. Since WebDriver is an interface, we can implement all methods of WebDriver interface. We cannot create an object of the interface directly. `WebDriver driver = new ChromeDriver();`
Step6:	Open URL of webpage in the browser by calling get() command. It takes a string URL as a parameter and returns nothing. The syntax is as follows: `driver.manage().timeouts().implicitlyWait(60, TimeUnit.SECONDS);`
Step7:	We will wait for the whole page to be load in browser by using the following syntax.
Step8:	To close browser, we will call close () command by using a reference variable driver. The syntax is as follows: `driver.close();`

Let us understand in detail the following program source code to automate the above scenarios.

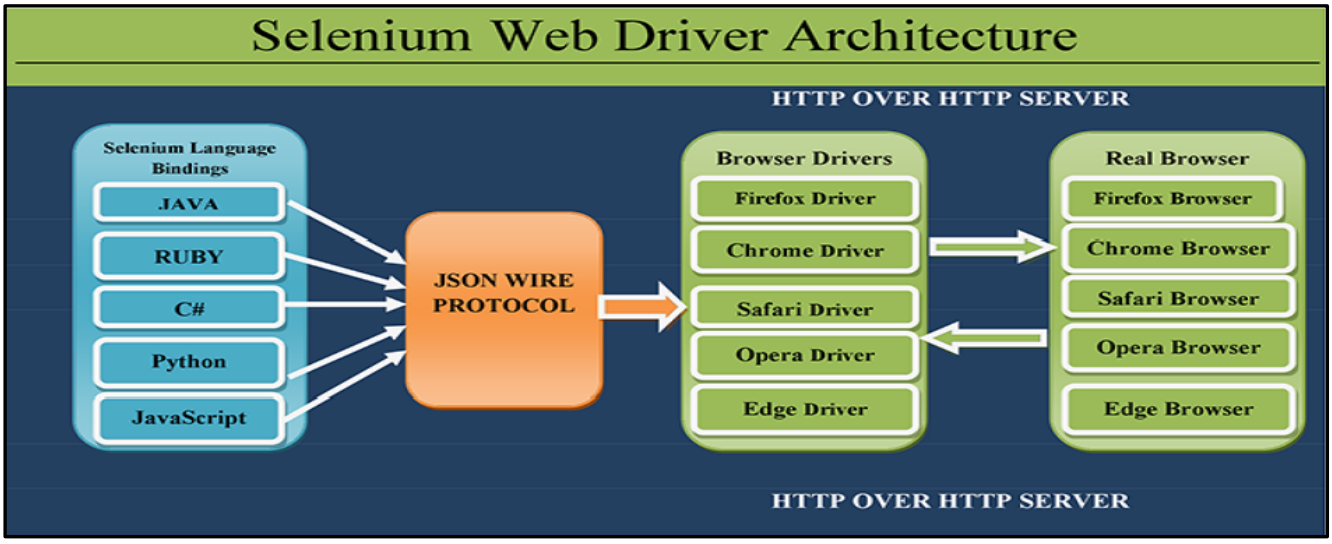
```java
package programs;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class OpenGoogleURL
{
    public static void main(String[] args) throws InterruptedException
    {
        System.setProperty("webdriver.chrome.driver","C:\\Selenium 2023\\chromed
        WebDriver driver = new ChromeDriver();
        String URL = "https://www.google.co.in";
        Thread.sleep(2000);
        driver.get(URL);
        Thread.sleep(2000);
        driver.manage().window().maximize();
        Thread.sleep(2000);
        System.out.println("Website is opened successfully");
        Thread.sleep(2000);
        driver.close();
    }
    }
```

```
Starting ChromeDriver 113.0.5672.63 (0e1a4471d5ae5bf128b1bd8f4d627c8cbd55f70c-refs/h
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions
ChromeDriver was started successfully.
May 23, 2023 12:10:34 AM org.openqa.selenium.devtools.CdpVersionFinder findNearestMa
WARNING: Unable to find an exact match for CDP version 113, so returning the closest
Website is opened successfully
```

Architecture of Selenium Webdriver:

WebDriver makes direct calls to the Web browser and the entire test script is executed in this fashion. WebDriver uses the browsers support and capabilities to automation. WebDriver is a purely object-oriented framework that works on OS layer. It utilizes the browser's native compatibility to automation without using any peripheral entity. With the increasing demand, it has gained a large popularity, user base and has become by far one of the most extensively used open-source automation testing tools. Selenium WebDriver is not a standalone testing tool. It comprises various components that are required to run tests. These are the architectural components of Selenium. This image tells us about the core selenium web driver architecture and the major selenium components which comprise WebDriver.

Understanding of Selenium WebDriver Architecture:



1. Selenium WebDriver Client Libraries:

Selenium supports multiple libraries such as Java, Ruby, Python, etc. It means that we can write our code with any of these scripting/programming languages. Software Testers want to select languages that they are comfortable with. Selenium Client Library sends the request in the form of API to the Browser Driver with the help of JSON Wire Protocol over HTTP (hypertext Transfer Protocol).

2. JSON Wire Protocol over HTTP:

JSON stands for JavaScript Object Notation. It is a lightweight data-interchange format which transfers the data between a server and a client on the web. The JSON Wire Protocol facilitates all the communication that is happening in Selenium between the browser and the code. JSON Wire Protocol is a REST API that transfers the data between HTTP server. Every Browser Driver uses a HTTP server to receive HTTP requests.

3. Browser Drivers:

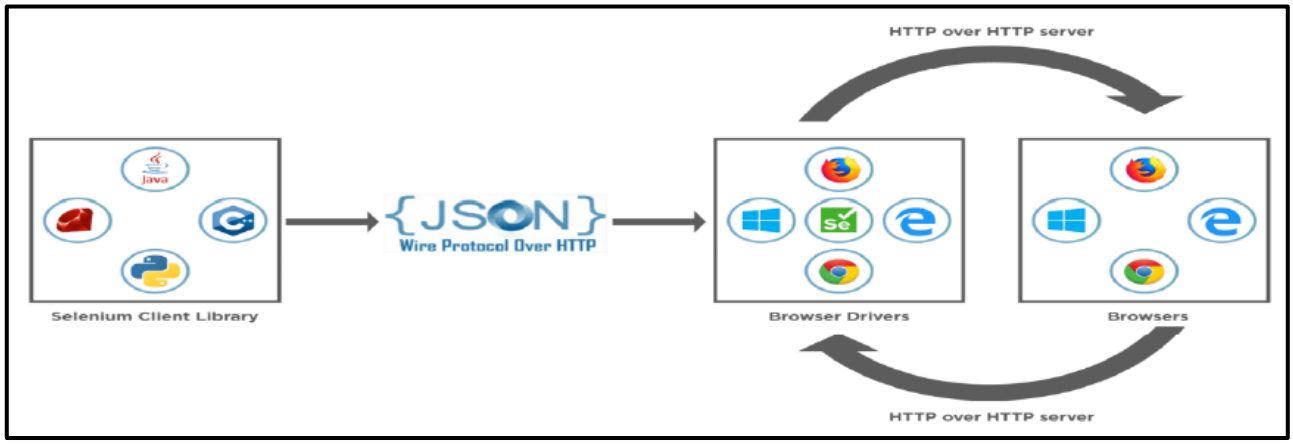
Browser Drivers are used to communicate with browsers. Since there are various browsers that are supported by Selenium. Each browser has its specific Browser WebDriver. When a browser driver is received any command then that command will be executed on the respective browser and the response will go back in the form of HTTP response. JSON Wire protocol establishes a connection between the browser binaries and the client libraries.

Following are the operations performed when we run our automation script using specific Browser driver:

a) For each Selenium command, a HTTP () request is created and sent to the browser driver.
b) The browser driver uses a HTTP server for getting the HTTP requests.
c) HTTP Server sends all the steps to perform a function which are executed on the browser.
d) The HTTP server sends the status back to the automation script.

4. Browsers:

Selenium will be only able to run tests on the browsers if they are locally installed, either on the local machine or on the server machines. So, browser installation is necessary. Selenium supports multiple browsers such as Firefox, Chrome, IE, Safari etc.



Now let's proceed to see the real implementation of the Same:

Here, the Selenium client library is your friend who knows the directions, the test script is the tourist and the web driver is you. You interacted with the tourist with your multilingual skills (let's say this is the Selenium API) and successfully executed the script, that is, our tourist got their directions and reached their destination.

```java
package com.example.testAutomation;

import org.openqa.selenium.WebDriver;

/**
 * @author Mandeep Kaur
 * @Date 17 April,2020
 */
public class SeleniumConfig {
    public static void main(String[] args) {

        String path = System.getProperty("user.dir");
        System.setProperty("webdriver.chrome.driver", path + "/chromedriver");
        WebDriver driver = new ChromeDriver();
        driver.get("https://www.hubspot.com/");

    }
}
```

WebDriver's main feature is to control and provide instructions to the Browser. So, you can choose any driver based on your browser. In above case we have chosen ChromeDriver.

Introduction WebDriver:

Selenium WebDriver is the successor to Selenium RC which sends commands directly to the browser and retrieves results. WebDriver is a tool for automating testing web applications. It is popularly known as Selenium 2.0. WebDriver uses a different underlying framework, while Selenium RC uses JavaScript

Selenium-Core embedded within the browser which has got some limitations. WebDriver interacts directly with the browser without any intermediary, unlike Selenium RC that depends on a server. The WebDriver proves to be better than Selenium IDE and Selenium RC in many aspects. It implements a more modern and stable approach in automating the browser's actions. Selenium WebDriver is a browser automation framework that accepts commands and sends them to a browser. It is implemented through a browser-specific driver. It directly communicates with the browser and controls it. Selenium WebDriver supports various programming languages like – Java, C#, PHP, Python, Perl, Ruby. and Javascript. WebDriver directly calls the methods of different browsers hence we have separate driver for each browser. **Some of the most widely used web drivers include:**

- Mozilla Firefox Driver (Gecko Driver)
- Google Chrome Driver
- Internet Explorer Driver
- Opera Driver
- HTML Unit Driver (Headless Driver)

Objects in Selenium:

While writing scripts for testing a web application using any automation tools, the tool should identify the web elements or web objects appropriately to perform the desired operation like click or enter text on the corresponding web element/object. This is called object identification. Every application is made of various objects like links, button, text box, drop down selector, radio button, list boxes, sliders, etc.

Accurate object identification is a very critical part of automating your application as only if you identify the right object, rest of your script runs without throwing errors. Tools like QTP and RFT which have record and playback features have mechanisms to store the properties of the GUI objects while recording and recognize and identify these objects later during run time. These tools also have features like 'Object spy' which helps you capture properties of objects you want to use in your tests.

Object identification in Selenium test automation is important. It helps you handle a dynamic list of object id changes, based on the number of rows. It also provides better validation options for the list of objects in one screen. It helps you find elements where the relative XPath is different across devices. And, finally, it helps you get dynamic data from the screen.

Object Identification in Selenium:

Selenium Webdriver interacts with webpage through object identification. Identification of any web element is performed by reading its source code and finding the unique value of the defined web element. Identification

of object in selenium performed using findElement () and findElements (). In Selenium, however, object identification is the responsibility of the automation tester writing scripts.

Here's how to identify objects in Selenium. Your first step is to get an object list. In Selenium, you can get list of objects with specific property. Then you can manipulate it in the code.

- ✓ Find the Common Property
- ✓ Use findElements Command
- ✓ Loop Over the WebElement List

****

Launching Chrome, Microsoft Edge and Mozilla FireFox Browsers

I. ChromeDriver:

A ChromeDriver is a standalone server or a separate executable used by Selenium WebDriver to control Chrome. Running Selenium test scripts on the Google Chrome browser is impossible without ChromeDriver. ChromeDriver is a separate executable that Selenium WebDriver uses to control Chrome. One can easily initialize the object of ChromeDriver using the following command:

WebDriver driver = new ChromeDriver

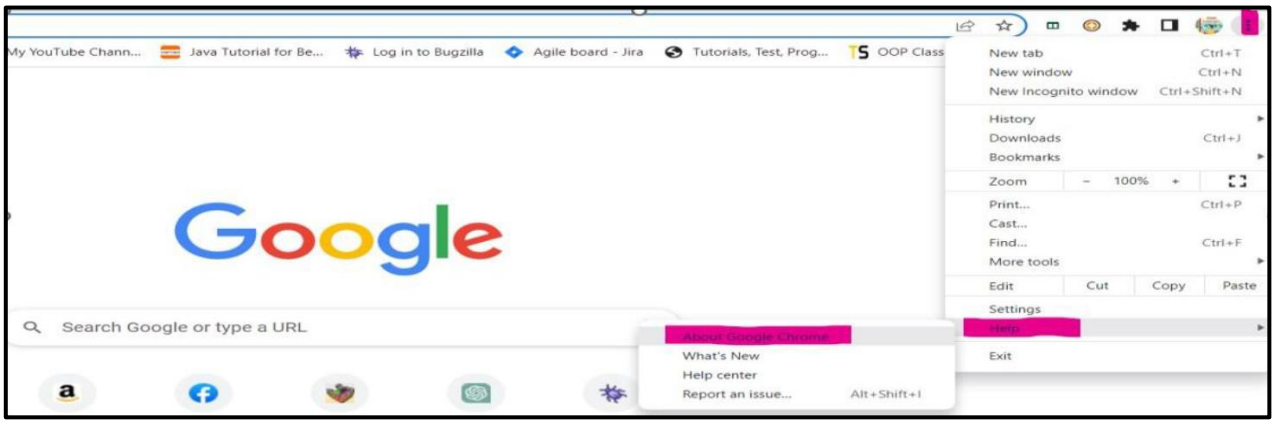
We assume that you have already installed the Google Chrome browser. The next step is to find the appropriate version of the ChromeDriver. Chromedriver is a .exe file that your WebDriver interface uses to initiate the Google Chrome browser. As this is an open tool, you can download it from its official website or the Selenium community. The only point that you need to consider is that the version of your Chrome browser should be compatible with the chromedriver.exe that you are going to download.

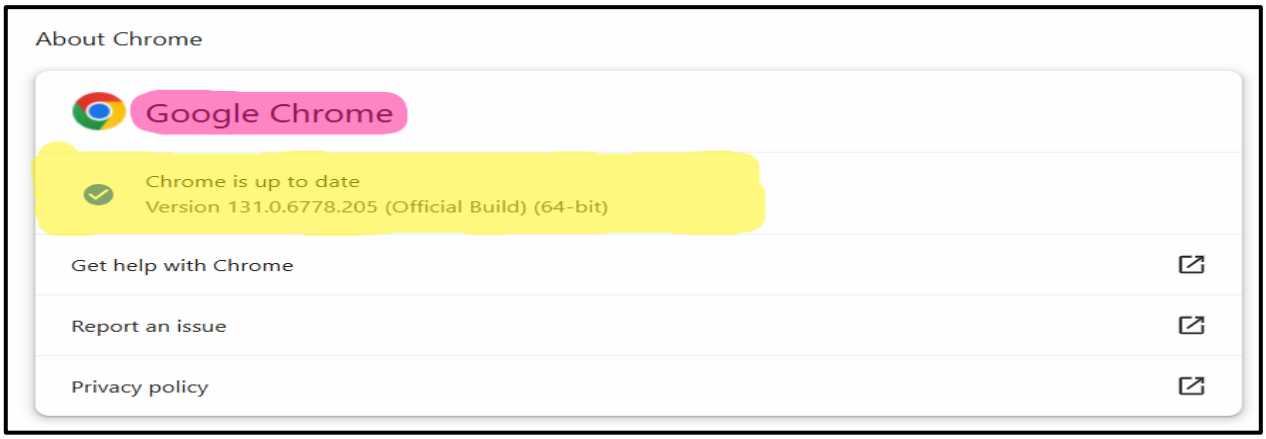
Download, Setup and Integration of Selenium Edge Driver:

Below are the steps to follow while configuring the chrome setup for Selenium:

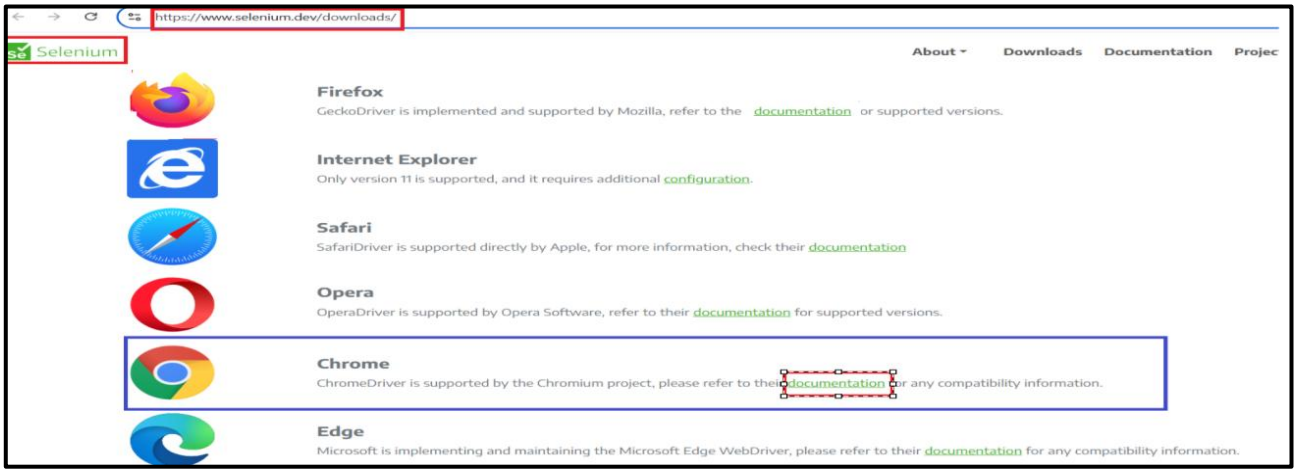
Step#1: Check the version of the chrome.

Open Chrome Browser -> Help -> About Google Chrome





Step#2: Users first need to download ChromeDriver for their respective Operating systems from this link [https://developer.chrome.com/docs/chromedriver or https://www.selenium.dev/downloads] you will see the latest ChromeDriver for the latest google chrome version.

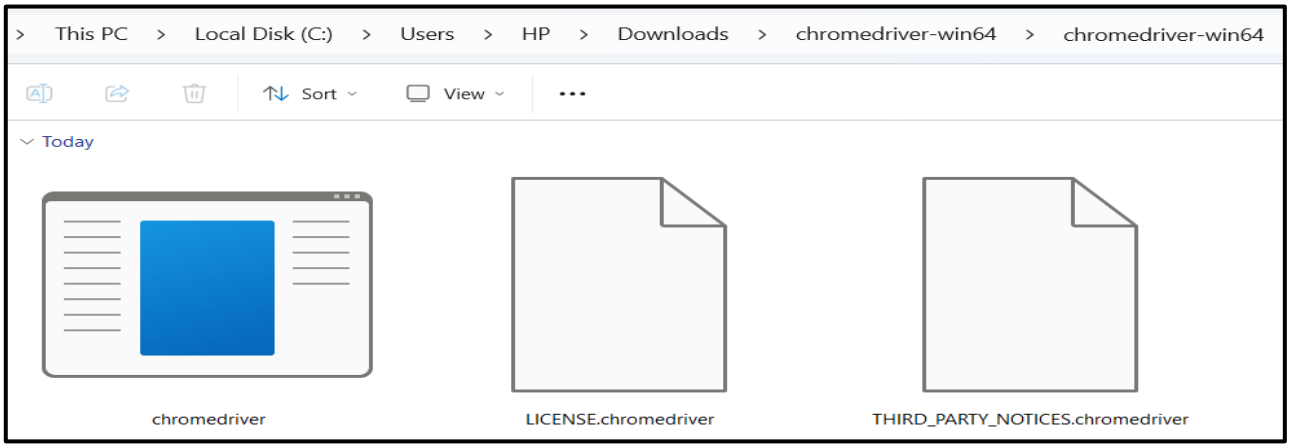


Stable

Version: 131.0.6778.204 (r1368529)

Binary	Platform	URL
chrome	linux64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.204/linux64/chrome-linux64.zip
chrome	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.204/mac-arm64/chrome-mac-arm64.zip
chrome	mac-x64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.204/mac-x64/chrome-mac-x64.zip
chrome	win32	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.204/win32/chrome-win32.zip
chrome	win64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.204/win64/chrome-win64.zip
chromedriver	linux64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.204/linux64/chromedriver-linux64.zip
chromedriver	mac-arm64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.204/mac-arm64/chromedriver-mac-arm64.zip
chromedriver	mac-x64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.204/mac-x64/chromedriver-mac-x64.zip
chromedriver	win32	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.204/win32/chromedriver-win32.zip
chromedriver	win64	https://storage.googleapis.com/chrome-for-testing-public/131.0.6778.204/win64/chromedriver-win64.zip

Step#3: Once the zip file is downloaded for the operating system, unzip it to retrieve the chromedriver.exe executable file. Copy this file to a specific location of your choice.



Step#4: We will jump into the setup and integration of Chrome driver with the Selenium framework to automate the Microsoft Edge browser. The path of the chromedriver ("C:\\Users\\HP\\Downloads\\chromedriver-win64\\chromedriver-win64\chromedriver.exe") will be used in our program.

Step#5: Finally, we will integrate the Chrome Driver with the Selenium framework to open a browser session.

Let us consider a test case in which we will try to automate the following scenarios in Google Chrome browser [Chrome Driver]

→ Launch the Chrome browser.

→ Maximize the browser.
→ Open URL: https://www.google.co.in.
→ Wait for the page to load.
→ Print a message (URL is opened successfully) on the console.
→ Close the browser.

```
1  package Browser_Drivers;
2  import org.openqa.selenium.WebDriver;
3  import org.openqa.selenium.chrome.ChromeDriver;
4  public class ChromeDriver_ChromeBrowser
5  {
6      public static void main(String[] args) throws InterruptedException
7      {
8          System.setProperty("webdriver.chrome.driver", "C:\\Users\\HP\\Downloads\\chromedri
9          WebDriver driver = new ChromeDriver();
10         String URL = "https://www.google.co.in/";
11         Thread.sleep(2000);
12         driver.get(URL);
13         Thread.sleep(2000);
14         driver.manage().window().maximize();
15         Thread.sleep(2000);
16         System.out.println("Website is opened successfully");
17         Thread.sleep(2000);
18         driver.close();
19             }
20     }
```

```
eclipse-workspace-2024-09 - SELENIUM_6PM_NOV24_BATCH/src/Browser_Drivers/ChromeDriver_ChromeBrowser.java - Eclipse IDE
File  Edit  Source  Refactor  Navigate  Search  Project  Run  Window  Help
Console ×
<terminated> ChromeDriver_ChromeBrowser [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe  (04-Jan-2025, 9:54:13 am – 9:54:26 am) [pid: 17844]
                        org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find CDP implementation matching 131
                        org.openqa.selenium.chromium.ChromiumDriver lambda$new$5
WARNING: Unable to find version of CDP to use for 131.0.6778.205. You may need to include a
Website is opened successfully
```

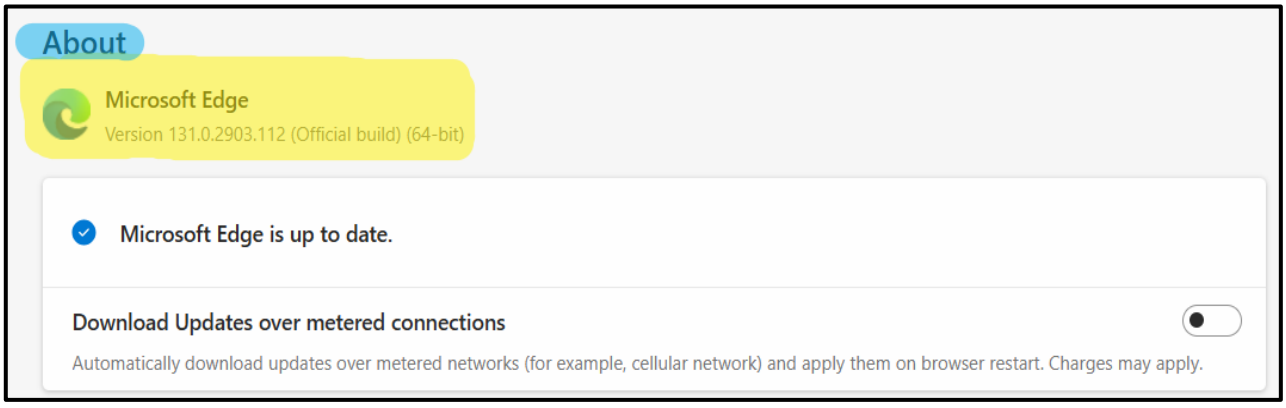
II. Microsoft-Edge Driver:

Microsoft Edge has been the trending news in the last couple of months due to its growing AI capabilities. Like all other popular browser vendors, Microsoft also provides a separate and specialized driver named "Edge Driver." Microsoft offers a Selenium WebDriver for automating the Microsoft Edge browser called Edge Driver. It acts as a bridge for seamless communication between the Selenium framework and the Microsoft Edge browser. Whether it be x86, x64 Windows, Mac, or Linux, the Edge driver comes with different versions to support all of them. Any version can be downloaded based on the browser version and the operating system we want to use.

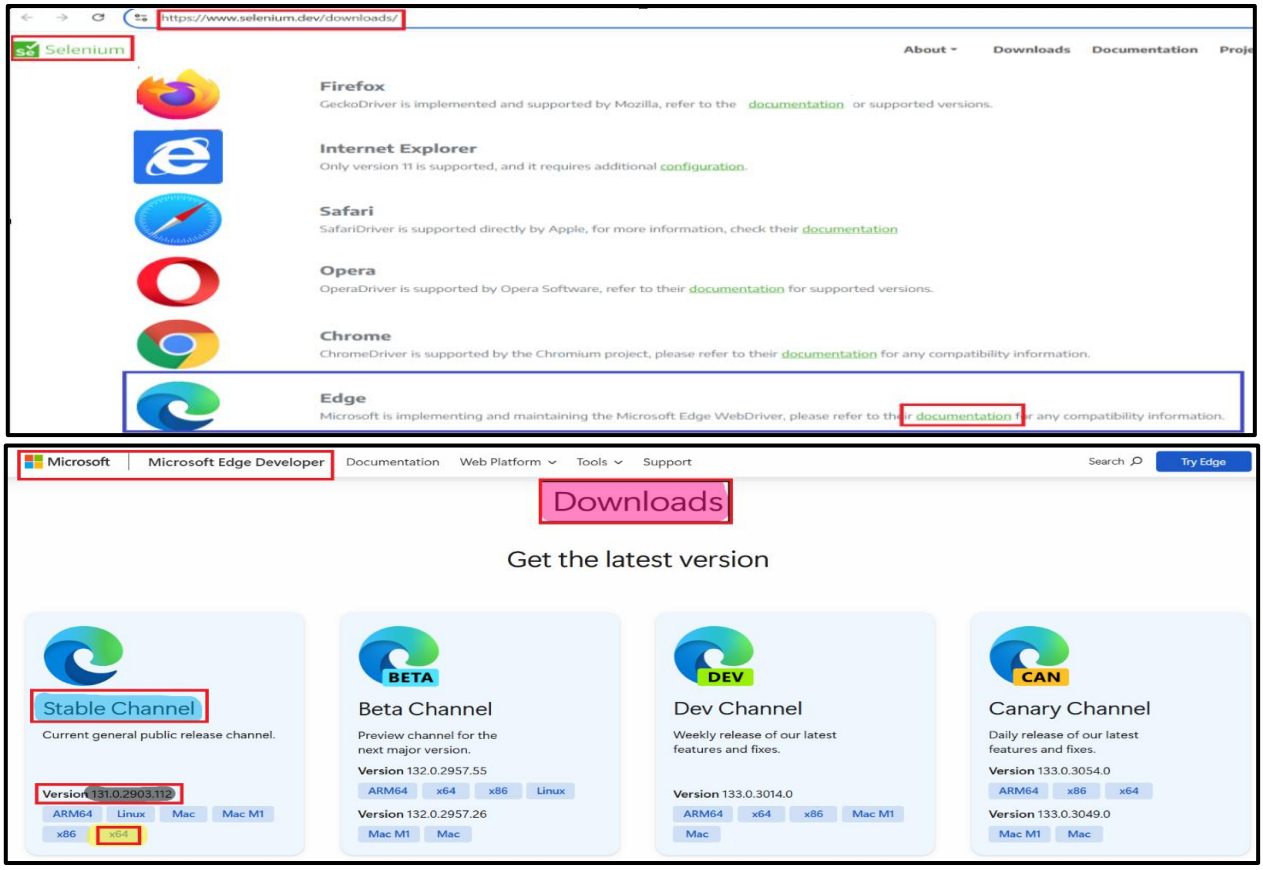
WebDriver driver = new EdgeDriver

Download, Setup and Integration of Selenium Edge Driver:

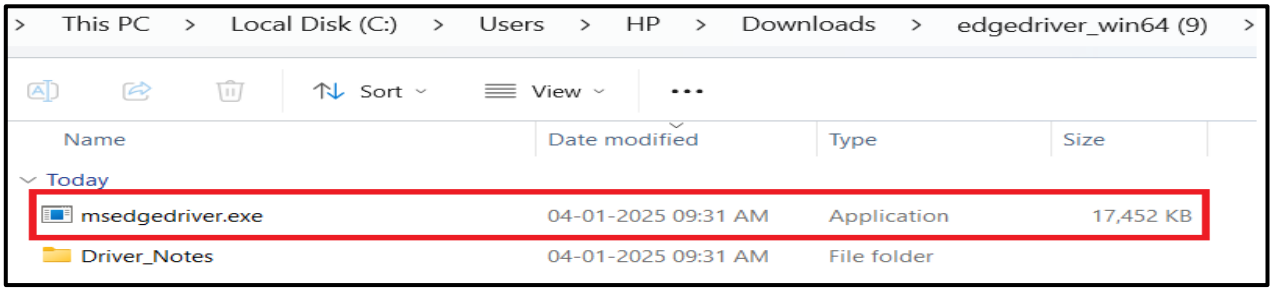
Step#1: First, we need to check the current version of the Microsoft Edge browser. To do so, open the Edge browser and click on "Settings and More" (the three dots) at the top right corner or press alt + F. Then, hover over "Help and Feedback" in the settings menu and click on About Microsoft Edge. On the About page, we can find the current browser version, as shown in the image below.



Step#2: Next, we need to download the Edge driver for its setup and integration. We can visit the official Microsoft Edge driver download page to download the Edge driver. On the page, we can select the appropriate version based on the operating system and browser to download it. Make sure to download the correct version according to the browser; otherwise, we may get a runtime error or some unexpected crash.



Step#3: Once the download is complete, a zip file gets saved. We need to extract the downloaded zip file and store it in the desired location. After extracting it, we can find the executable file – msedgedriver.exe file at the selected location as sown in the image below:

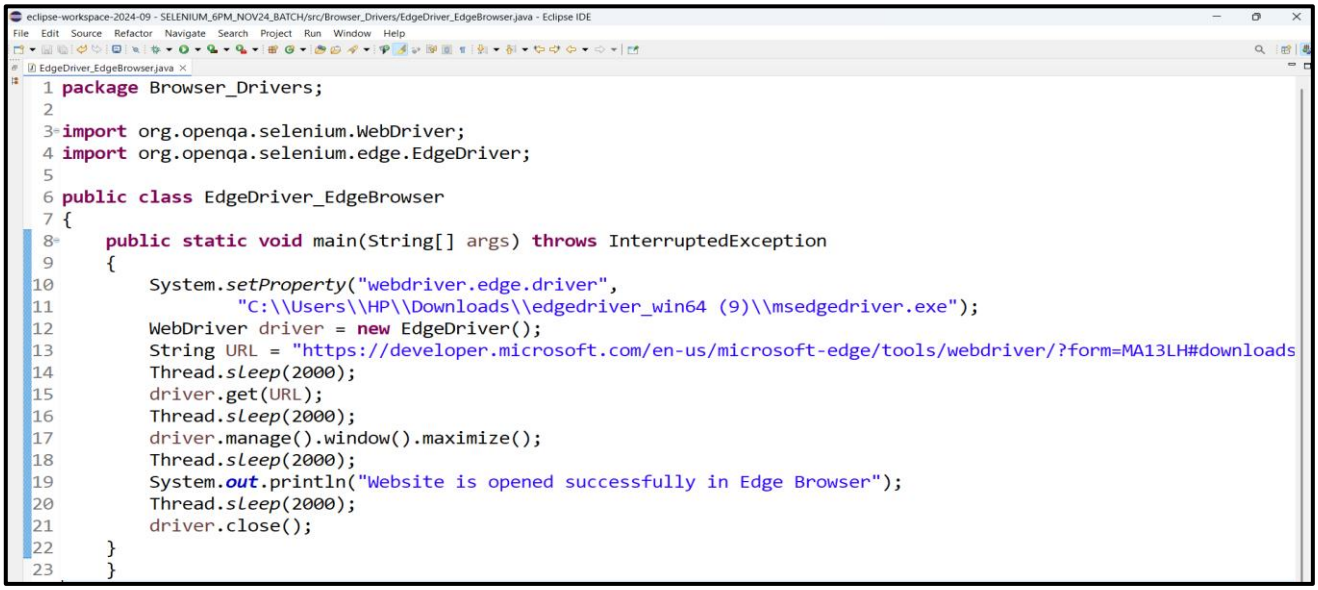


Step#4: We will jump into the setup and integration of Edge driver with the Selenium framework to automate the Microsoft Edge browser. The path of the edgedriver "C:\\Users\\HP\\Downloads\\edgedriver_win64 (9)\\msedgedriver.exe") will be used in our program.

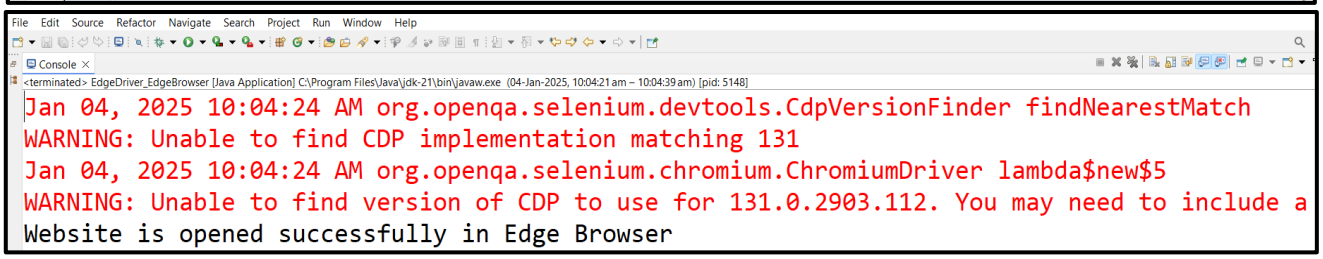
Step#5: Finally, we will integrate the Edge driver with the Selenium framework to open a browser session. Here we will open the browser and then navigate to a web page with an automation script.

Let us consider a test case in which we will try to automate the following scenarios in Microsoft-Edge Browser [Edge Driver]

- → Launch the Chrome browser.
- → Maximize the browser.
- → Open URL: https://www.google.co.in.
- → Wait for the page to load.
- → Print a message (URL is opened successfully) on the console.
- → Close the browser.



```
package Browser_Drivers;

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

public class EdgeDriver_EdgeBrowser
{
    public static void main(String[] args) throws InterruptedException
    {
        System.setProperty("webdriver.edge.driver",
                "C:\\Users\\HP\\Downloads\\edgedriver_win64 (9)\\msedgedriver.exe");
        WebDriver driver = new EdgeDriver();
        String URL = "https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/?form=MA13LH#downloads
        Thread.sleep(2000);
        driver.get(URL);
        Thread.sleep(2000);
        driver.manage().window().maximize();
        Thread.sleep(2000);
        System.out.println("Website is opened successfully in Edge Browser");
        Thread.sleep(2000);
        driver.close();
    }
    }
```



```
Jan 04, 2025 10:04:24 AM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find CDP implementation matching 131
Jan 04, 2025 10:04:24 AM org.openqa.selenium.chromium.ChromiumDriver lambda$new$5
WARNING: Unable to find version of CDP to use for 131.0.2903.112. You may need to include a
Website is opened successfully in Edge Browser
```

III. Firefox Driver / GeckoDriver:

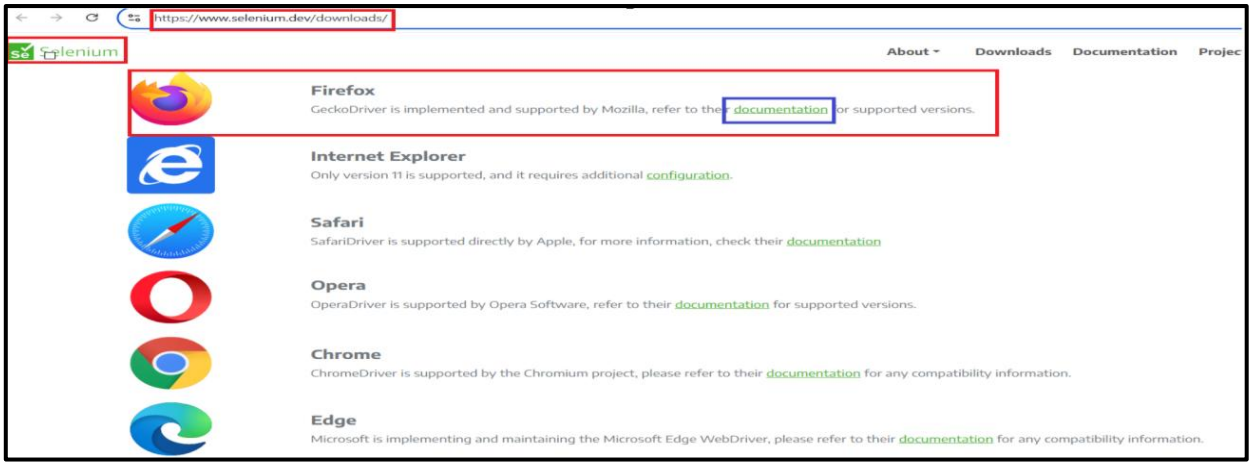
Selenium Firefox Driver, also called GeckoDriver, is a browser rendering engine developed by Mozilla for many applications. It provides a link between test cases and the Firefox browser. Without the help of GeckoDriver, one cannot instantiate the object of the Firefox browser and perform automated Selenium testing. The term Gecko stands for a Web Browser engine that is inbuilt within Mozilla Firefox browser. Gecko driver acts as a proxy between Web Driver enabled clients(Eclipse, Netbeans, etc.) and Mozilla Firefox browser.

In short, Gecko driver acts as a link between Selenium Web Driver tests and Mozilla Firefox browser.

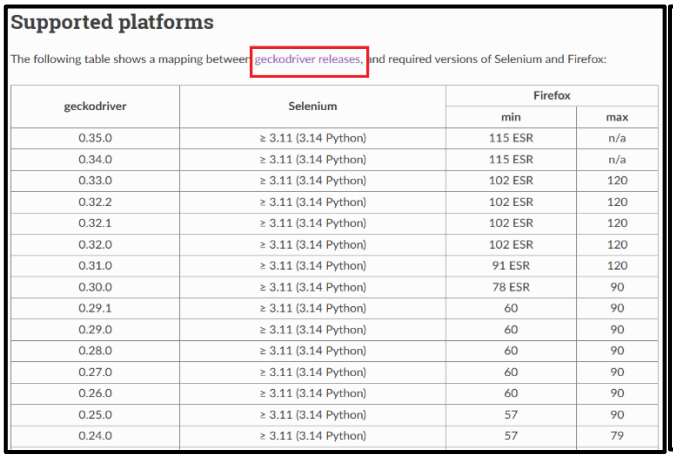
Download, Setup and Integration of Selenium Firefox Driver:

Step#1: First, we need to check the current version of the Mozilla Firefox browser. To do so, open the Firefox browser and click the menu button (three horizontal lines) in the top right corner and Select Help and Select About Firefox. We can find the current browser version, as shown in the image below.

Step#2: Next, we need to download the Firefox driver for its setup and integration. Navigate to the official Selenium website. Under third-party drivers, one will find all the drivers. Just click on the Mozilla GeckoDriver documentation, as shown below.



Step#3: After that, check the latest supported platforms of GeckoDriver versions in the documentation and click on GeckoDriver releases as shown below. Now, it will navigate to the GeckoDriver downloads link, where one can download the suitable driver based on the OS as it is platform agnostic. The snapshot below depicts all the available Selenium Firefox Driver releases.



Supported platforms

The following table shows a mapping between geckodriver releases and required versions of Selenium and Firefox:

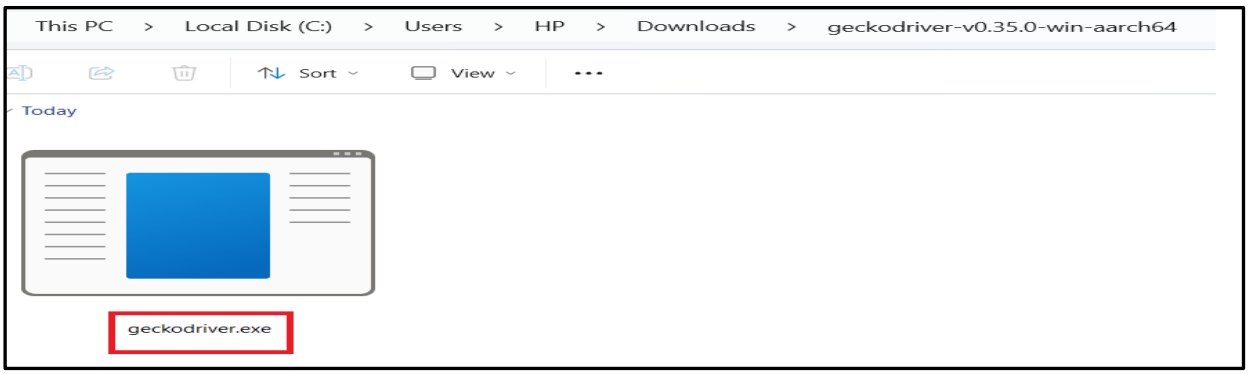
geckodriver	Selenium	Firefox	
		min	max
0.35.0	≥ 3.11 (3.14 Python)	115 ESR	n/a
0.34.0	≥ 3.11 (3.14 Python)	115 ESR	n/a
0.33.0	≥ 3.11 (3.14 Python)	102 ESR	120
0.32.2	≥ 3.11 (3.14 Python)	102 ESR	120
0.32.1	≥ 3.11 (3.14 Python)	102 ESR	120
0.32.0	≥ 3.11 (3.14 Python)	102 ESR	120
0.31.0	≥ 3.11 (3.14 Python)	91 ESR	120
0.30.0	≥ 3.11 (3.14 Python)	78 ESR	90
0.29.1	≥ 3.11 (3.14 Python)	60	90
0.29.0	≥ 3.11 (3.14 Python)	60	90
0.28.0	≥ 3.11 (3.14 Python)	60	90
0.27.0	≥ 3.11 (3.14 Python)	60	90
0.26.0	≥ 3.11 (3.14 Python)	60	90
0.25.0	≥ 3.11 (3.14 Python)	57	90
0.24.0	≥ 3.11 (3.14 Python)	57	79

Assets 13

geckodriver-v0.35.0-linux-aarch64.tar.gz	2.04 MB	Aug 6, 2024
geckodriver-v0.35.0-linux-aarch64.tar.gz.asc	833 Bytes	Aug 6, 2024
geckodriver-v0.35.0-linux32.tar.gz	2.09 MB	Aug 6, 2024
geckodriver-v0.35.0-linux32.tar.gz.asc	833 Bytes	Aug 6, 2024
geckodriver-v0.35.0-linux64.tar.gz	2.15 MB	Aug 6, 2024
geckodriver-v0.35.0-linux64.tar.gz.asc	833 Bytes	Aug 6, 2024
geckodriver-v0.35.0-macos-aarch64.tar.gz	1.91 MB	Aug 6, 2024
geckodriver-v0.35.0-macos.tar.gz	2.11 MB	Aug 6, 2024
geckodriver-v0.35.0-win-aarch64.zip	1.54 MB	Aug 6, 2024
geckodriver-v0.35.0-win32.zip	1.62 MB	Aug 6, 2024
Source code (zip)		Aug 6, 2024
Source code (tar.gz)		Aug 6, 2024

Show all 13 assets

Step#4: Once the download is complete, a zip file gets saved. We need to extract the downloaded zip file and store it in the desired location. After extracting it, we can find the executable file – geckodriver.exe file at the selected location as sown in the image below:



Step#5: We will jump into the setup and integration of geckodriver with the Selenium framework to automate the Mozilla FireFox browser. The path of the geckodriver " "C:\\Users\\HP\\Downloads\\geckodriver-v0.35.0-win-aarch64\\geckodriver.exe" ") will be used in our program.

Step#6: Finally, we will integrate the Gecko driver with the Selenium framework to open a browser session. Here we will open the browser and then navigate to a web page (https://www.lambdatest.com) with an automation script.

Let us consider a test case in which we will try to automate the following scenarios in FireFox Browser [Gecko Driver]

- → Launch the FireFox browser.
- → Maximize the browser.
- → Open URL: https://www.google.co.in.
- → Wait for the page to load.
- → Print a message (URL is opened successfully) on the console.
- → Close the browser.

```java
package Browser_Drivers;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class FireFoxDriver_FireFoxBrowser
{
    public static void main(String[] args) throws InterruptedException
    {
        System.setProperty("webdriver.firefox.driver",
                "C:\\Users\\HP\\Downloads\\geckodriver-v0.35.0-win-aarch64\\geckodriver.exe");
        WebDriver driver = new FirefoxDriver();
        String URL = "https://firefox-source-docs.mozilla.org/testing/geckodriver/Support.html";
        Thread.sleep(2000);
        driver.get(URL);
        Thread.sleep(2000);
        driver.manage().window().maximize();
        Thread.sleep(2000);
        System.out.println("Website is opened successfully in Mozilla Browser");
        Thread.sleep(2000);
        driver.close();
    }
}
```

```
Website is opened successfully in Mozilla Browser
```

Browser	Syntax
Chrome	WebDriver driver = new ChromeDriver();
Edge	WebDriver driver = new EdgeDriver();
Firefox	WebDriver driver = new FirefoxDriver();
Safari	WebDriver driver = new SafariDriver();

Difference Between WebDriver, ChromeDriver, EdgeDriver, and GeckoDrive in Selenium:

WebDriver	ChromeDriver	EdgeDriver	GeckoDrive (Firefox)
WebDriver is a generic interface in Selenium that provides a way to interact with web browsers.	ChromeDriver is a specific implementation of WebDriver for the Google Chrome browser.	EdgeDriver is a specific implementation of WebDriver for the Microsoft Edge browser.	GeckoDriver is a specific implementation of WebDriver for the Mozilla Firefox browser.
Can be used with various browsers by initializing the appropriate driver (e.g., ChromeDriver, FirefoxDriver).	Primarily used for automating tasks in the Google Chrome browser.	Primarily used for automating tasks in the Microsoft Edge browser.	Primarily used for automating tasks in the Mozilla Firefox browser.

Supports multiple browsers such as Chrome, Firefox, Edge, Safari, etc.	Specifically designed for the Chrome browser.	Specifically designed for the Edge browser.	Specifically designed for the Firefox browser.
Part of the Selenium project and is actively developed to support various browsers.	Developed by the Chromium project to support automation specifically for the Chrome browser.	Developed by the Microsoft Edge team to support automation specifically for the Edge browser.	Developed by the Mozilla project to support automation specifically for the Firefox browser.
Can be used with various browsers by initializing the appropriate driver (e.g., ChromeDriver, FirefoxDriver).	Need to instantiate the ChromeDriver class.	Need to instantiate the EdgeDriver class.	Need to instantiate the FirefoxDriver class.
WebDriver driver = new ChromeDriver(); WebDriver driver = new EdgeDriver(); WebDriver driver = new FirefoxDriver();	ChromeDriver driver = new ChromeDriver();	EdgeDriver driver = new EdgeDriver();	FirefoxDrive driver = new FirefoxDriver();
Updates are part of the Selenium releases, and WebDriver is versioned accordingly.	ChromeDriver updates are closely related to Chrome browser versions, and it's recommended to use compatible versions.	EdgeDriver updates are closely related to Microsoft Edge browser versions. It's recommended to use compatible versions.	GeckoDriver updates are closely related to Firefox browser versions, and it's recommended to use compatible versions.

****

Headless Browser Testing with Selenium

Selenium is one of the powerful web automation test suites to automate the testing of web applications against browsers such as Chrome, Firefox, IE, Edge, etc. It is one of the popular browser automation tools to verify website functioning, check website navigation flow, verify page titles, and fetch huge amounts of data by web scraping.

Along with all these, it offers one more fantastic feature: Headless Browser testing. Headless browser testing is nothing but running the Selenium tests on headless browsers. A headless browser is just like a real browser with no User Interface. Selenium also supports headless versions of real browsers like Chrome, Firefox, and Edge. Headless is an execution mode for Firefox and Chromium based browsers. It allows users to run automated scripts in headless mode, meaning that the browser window wouldn't be visible.

Headless execution is important because it allows developers and testers to perform automated tests and web scraping tasks without the need for a graphical user interface. In addition, headless execution is crucial since it facilitates CI/CD workflows, which stand for continuous integration and continuous delivery. It is necessary to release new code modifications rapidly and confidently in contemporary software development. It is simpler to integrate automated tests into the CI/CD pipeline by using headless testing, which enables them to execute rapidly and effectively in a headless environment.

Run Selenium Test Cases in Headless Mode:

There are **4** ways to run Selenium tests in headless mode let's understand them in the following sections:

1) **Running Selenium test cases using HTMLUnitDriver.**
2) **Running Selenium test cases using a headless Chrome browser.**
3) **Running Selenium test cases using the headless Firefox browser.**
4) **Running Selenium test cases using a headless Edge browser.**

1) Running Selenium test cases using HTMLUnitDriver:

HtmlUnitDriver is a Selenium WebDriver framework tool that allows you to test web applications automatically. It's designed for testing websites without even opening a visible browser. It uses a Java library called HtmlUnit, which imitates a browser's behavior.

Step#1: Download the Necessary Files:

To begin, go to the website and download the files you need for HtmlUnitDriver. Ensure you get the latest version and save it on your computer. After that, open your project in Eclipse and add the downloaded file to your project's build path. To do this, right-click on your project, go to Build Path, then Configure Build Path. In the Libraries section, click Add External JARs, choose the file you downloaded from its folder, and click Apply and Close to complete the process.

Step#2: Check the Downloaded File:

After including the file, you can effortlessly import the required class to your test by executing the following command.

import org.openqa. Selenium.htmlunit.HtmlUnitDriver;

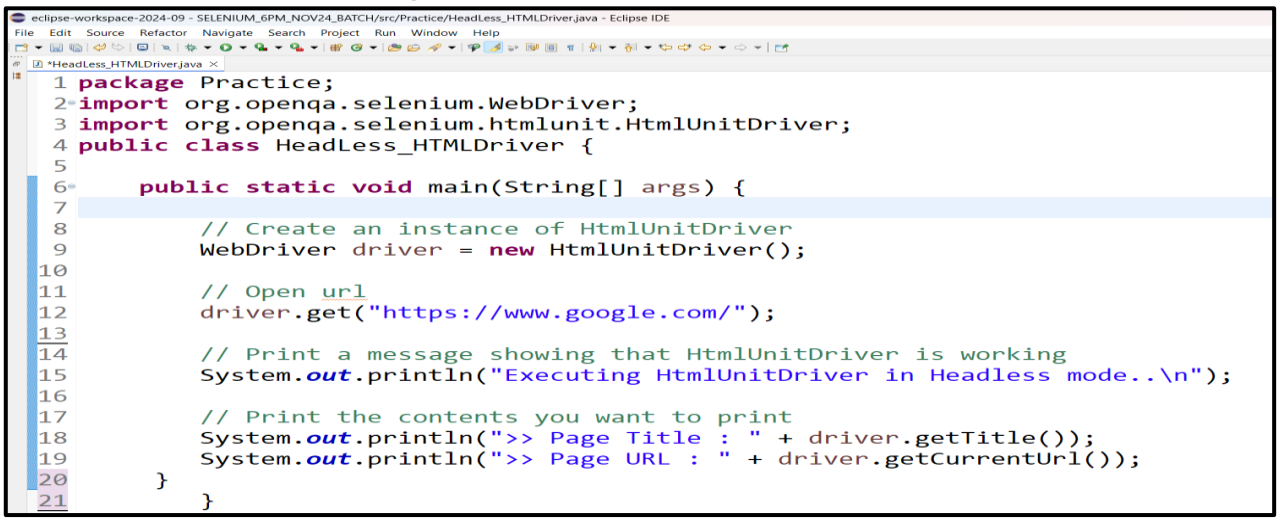
Step#3: Create a New Instance of HtmlUnitDriver:

We can create new instances using the following command.

WebDriver driver = new HTMLUnitDriver();

Step#4: Write The Test Code Using HTMLUnitDriver.

You can write a test code using the created HTMLUnitDriver instance.



2) Running Selenium test cases using a Headless Chrome Browser:

Headless Chrome is a version of the Google Chrome browser that works without a graphical interface. Headless Chrome is a way to run the Chrome browser in a headless environment without the full browser user interface. Headless Chrome offers you a real browser context without the memory overhead of running a full version of Chrome. Google Chrome is available with headless execution since version 59. Selenium WebDriver provides a class called "ChromeOptions", which can specify certain configurations to change the default behavior of Chrome. One of those configurations is the "headless" mode, which launches

the Chrome in headless mode while running the test cases. The following code snippet shows how we can pass the "headless" option using the ChromeOptions class. It also displays how we can then use those options while creating the instance of the ChromeDriver:

Step#1: Download and install the latest version of ChromeDriver.

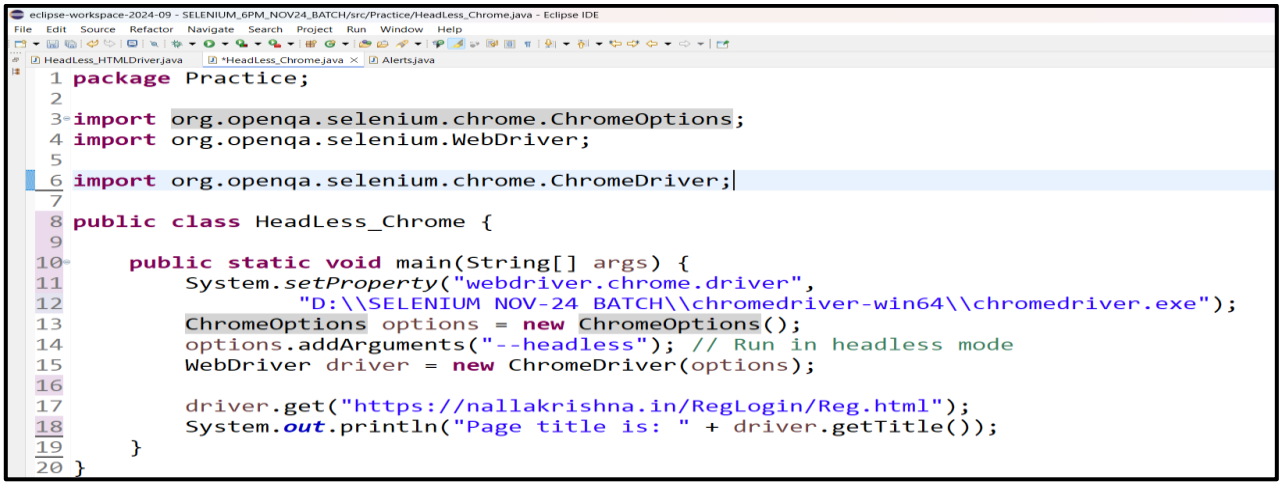
Step#2: Add the path to ChromeDriver to your system's PATH environment variable.

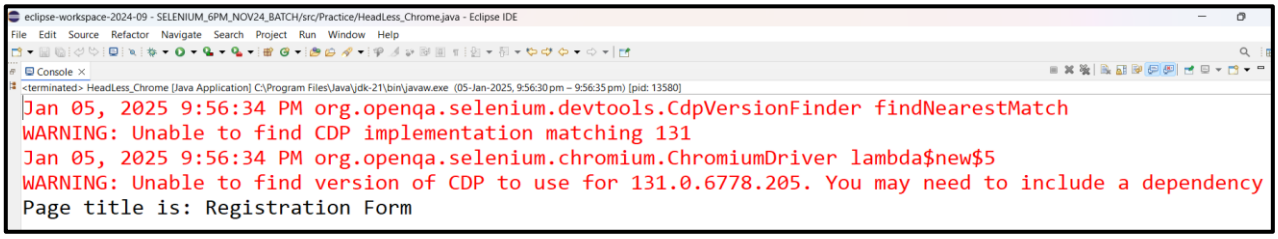
Step#3: Create a new instance of ChromeDriver in your Selenium code:

```
ChromeOptions options = new ChromeOptions();
options.addArguments("--headless");
WebDriver driver = new ChromeDriver(options);
```

Step#4: Set option.setHeadless() to true as shown in the code above

Step#5: Write your Selenium test code and run it using the ChromeDriver.





3) Running Selenium test cases using the Headless Firefox Browser:

Headless Firefox is a headless browser that can be used for running Selenium tests in headless mode. Here's how to use Headless Firefox to run Selenium tests:

Step 1: Download and install the latest version of geckodriver.

Step 2: Add the path to geckodriver to your system's PATH environment variable.

Step 3: Create a new instance of FirefoxDriver in your Selenium code.

```
FirefoxOptions options = new FirefoxOptions();
options.addArguments("--headless"); // Run in headless mode
WebDriver driver = new FirefoxDriver(options);
```

Step 4: Set option.setHeadless() to true as shown in the code above

Step 5: Write your Selenium test code and run it using the FirefoxDriver

```java
package Practice;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.firefox.FirefoxOptions;

public class HeadLess_FireFox {

    public static void main(String[] args) {
        System.setProperty("webdriver.gecko.driver",
                "D:\\SELENIUM NOV-24 BATCH\\geckodriver.exe");
        FirefoxOptions options = new FirefoxOptions();
        // options.setHeadless(true); // Enable headless mode
        options.addArguments("--headless"); // Run in headless mode
        WebDriver driver = new FirefoxDriver(options);
        driver.get("https://www.facebook.com/");
        System.out.println("Page title is: " + driver.getTitle());
    }
}
```

```
Page title is: Facebook - log in or sign up
```

4) Running Selenium test cases using a Headless Edge Browser:

Headless Edge is a Selenium headless browser that can be used for running Selenium tests in headless mode. Here's how to use Headless Edge to run Selenium tests:

Step#1: Download and install the latest version of Microsoft Edge Driver.

Step#2: Add the path to Microsoft Edge Driver to your system's PATH environment variable

Step#3: Create a new instance of EdgeDriver in your Selenium code:

EdgeOptions options = new EdgeOptions();
options.addArguments("--headless"); // Run in headless mode
WebDriver driver = new EdgeDriver(options);

Step#4: Set the headless flag in option.setCapability() to true as shown in the code above

Step#5: Write your Selenium test code and run it using the EdgeDriver

```java
package Practice;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.edge.EdgeDriver;
import org.openqa.selenium.edge.EdgeOptions;

public class HeadLess_Edge {

    public static void main(String[] args) {
        System.setProperty("webdriver.edge.driver",
                "D:\\SELENIUM NOV-24 BATCH\\msedgedriver.exe");
        EdgeOptions options = new EdgeOptions();
        // options.setHeadless(true); // Enable headless mode
        options.addArguments("--headless"); // Run in headless mode
        WebDriver driver = new EdgeDriver(options);

        driver.get("https://www.youtube.com/channel/UCFJPoD6sWwRsm8CYxGjoMeg");
        System.out.println("Page title is: " + driver.getTitle());
    }
}
```

```
Jan 05, 2025 10:14:54 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find CDP implementation matching 131
Jan 05, 2025 10:14:54 PM org.openqa.selenium.chromium.ChromiumDriver lambda$new$5
WARNING: Unable to find version of CDP to use for 131.0.2903.112. You may need to include a dependency
Page title is: Krishna Software Trainer - YouTube
```

****

Verify Page Title and URL in Selenium

Title Verification [getTitle()]:

When we are working with selenium, sometimes there comes a need to verify the page title. Generally, page title verification needs to be done when we work around a web application. We have to verify the page title, and we must do as we navigate to different pages. This may act as a checkpoint on where we currently are.

In the context of automation with Selenium, "title verification" typically refers to the process of checking whether the title of a web page matches an expected title. This is a common validation step in automated testing to ensure that the correct page has been loaded or that the expected behavior has occurred.

Its criticality could be understood when we have to go through different web pages in same URL or domain and page title of that AUT is changing. It may act as a checkpoint. So if we have multiple web pages in same domain then we can proceed further validation of the functionality after verifying its page title only.

We use **getTitle()** method to get the actual title of any web page. We can also use If-statement to compare actual and expected web page title.

Here are the steps in the code to verify title of the page:

Step 1: Use getTitle() and store the value in String

Example: String actualTitle = driver.getTitle();

Step 2: Declare expected title in string

Example: String expectedTitle =”Your expected title”;

Step 3: Now compare expected and actual.

Sample Script:

```
public static void main(String[] args) {
    WebDriver driver = new ChromeDriver();
    System.setProperty("webdriver.chrome.driver",
            "C:\\Users\\HP\\Downloads\\chromedriver-win64 (1)\\chromedriver-
    driver.manage().window().maximize();
    // Get the url
    driver.get("https://www.facebook.com/");

    String expectedTitle = "Facebook - log in or sign up";
    String actualTitle = driver.getTitle();
    System.out.println("Actual Title is: "+actualTitle);
    System.out.println("Expected Title is: " + expectedTitle);
    // Title verification page
    if (actualTitle.equalsIgnoreCase(expectedTitle))
    {
        System.out.println("Title matched ");
    }
    else
    {
        System.out.println("Title didn't matched ");
    }
```

```
Console ×
FaceBook_TittleVerification [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (04-Mar-2024, 12:56:02 pm) [pid: 3784]
Mar 04, 2024 12:56:06 PM org.openqa.selenium.devt
WARNING: Unable to find CDP implementation matchi
Mar 04, 2024 12:56:06 PM org.openqa.selenium.chro
WARNING: Unable to find version of CDP to use for
Actual Title is: Facebook - log in or sign up
Expected Title is: Facebook - log in or sign up
Title matched
```

URL Verification [getCurrentUrl()]:

A URL (Uniform Resource Locator) is a web address that specifies the location of a resource on the internet. The URL of an application typically refers to the web address or endpoint that allows users to access or interact with the application. The URL of an application can vary depending on the specific application and how it is hosted or deployed. For example, if the application is a web-based application hosted on a server, the URL may look something like this:

https://www.tronixtechs.com/ (or) https://nallakrishna.in/RegLogin/Info.html

In Selenium-Java, you can use the getCurrentUrl() method of the WebDriver interface to get the current URL of the web page. getCurrentUrl() is a method in Selenium WebDriver that retrieves the current URL of the web page opened in the browser. This method returns a String value representing the current URL of the web page. This method can be useful for verifying that the browser has navigated to the expected URL, or for capturing the current URL for later use in your test script.

```
// Verify the current URL matches the expected URL
String actualURL = driver.getCurrentUrl();
String expectedURL = "https://www.example.com/";
System.out.println("Actual Url is:"+ actualURL);
if (actualURL.equals(expectedURL))
{
 System.out.println("URL redirection successful");
} else {
 System.out.println("URL redirection failed");
}
```

Different Methods to open a URL in Selenium:

1. get(): This is the standard way to open a URL in Selenium. The get() method is the most commonly used method to open a URL in Selenium. It opens the specified URL in the current browser window or tab.

Ex: **driver.get("https://www.example.com");**

2. navigate().to(): An alternative way to open a URL using the navigate() interface. The navigate().to() method is also used to open a URL, but it provides
more flexibility than get(). It allows you to navigate back and forward through the browser history, and also to refresh the page.

Ex: **driver.navigate().to("https://www.example.com");**

3. JavaScript Executor: Using JavaScript Executor (JavascriptExecutor) in Selenium WebDriver is useful when get() or navigate().to() methods fail due to security restrictions or page load issues. Here's a deeper look at how and when to use it.

Ex: **JavascriptExecutor js = (JavascriptExecutor) driver;**
js.executeScript("window.location = 'https://www.example.com'");

****

Assertions in Selenium

The word Assert means to state a fact or belief confidently or forcefully. In Selenium, Asserts are validations or checkpoints for an application. Assertions state confidently that application behavior is working as expected. Asserts in Selenium validate the automated test cases that help testers understand if tests have passed or failed. The assertion is considered to be met if the actual result of an application matches with that of the expected result. For creating assertion we are going to use the Assert class provided by TestNG.

A simple example of an assert can be opening the web page under test, matching the Page Title with the expected title, and verifying if the required WebElements on the page are loaded or not. In case the Page URL is incorrect (i.e. does not match with the expected URL), an assert should be thrown since the test scenarios would definitely fail!

There are Two Types of Assertions:

1. Hard Assertions: Hard Assertions are ones in which test execution is aborted if the test does not meet the assertion condition. The test case is marked as failed. In case of an assertion error, it will throw the "java.lang.AssertionError" exception. When any assert statement fails this type of assertion throws an exception immediately and continues with the next test in the test suite. The test case will be immediately marked as Failed when a hard assertion condition fails.

A. Assert Equals: assertEquals, assertNotEquals	
assertEquals: This is used to compare expected and actual values in the selenium webdriver. The assertion passes with no exception whenever the expected and actual values are same. But, if the actual and expected values are not same then assert fails with an exception and the test is marked as failed. **Syntax:**Assert.assertEquals(actual,expected);	**assertNotEquals:** This is just the opposite of assertEquals. The assertion passes with no exception whenever the expected and actual values are not same. But, if the actual and expected values are same then assert fails with an exception and the test is marked as failed. **Syntax:**Assert.assertNotEquals(actual, expected, message);
driver.navigate().to("https://www.tronixtechs.com/");	driver.navigate().to("https://www.tronixtechs.com/");

String actualTitle = driver.getTitle(); String expectedTitle = "No.1 Python training in Hyderabad|Tronix Technologies"; assertEquals(actualTitle, expectedTitle, "Page title does not match the expected value");	String actualTitle = driver.getTitle(); String expectedTitle = "Incorrect Title"; assertNotEquals(actualTitle, expectedTitle, "Titles should not match, but they do");
B. Assert Boolean: assertTrue, assertFalse	
assertTrue: This type of assertion is used when you are checking if condition is true. That is when we are dealing with boolean values this assertion is used. Whenever test case passes it returns true and if condition is false then it skips the current method and jumps to next. **Syntax:** Assert.assertTrue(condition);	**assertFalse**: It checks if value returned is false or not. Whenever test case passes it aborts the method and gives an exception. **Syntax:** Assert.assertFalse(condition);

driver.navigate().to("https://www.tronixtechs.com/");
boolean verifyTitle = driver.getTitle().equalsIgnoreCase("No.1 Python training in Hyderabad|Tronix Technologies");
assertTrue(verifyTitle, "The page title is not as expected");
	driver.navigate().to("https://www.tronixtechs.com/");
	boolean verifyTitle = driver.getTitle().equalsIgnoreCase("Incorrect Title");
	assertFalse(verifyTitle, "The title matched an incorrect value");

C. Assert None: assertNull, assertNotNull	
assertNull: This assertion checks if the object is null or not. If an object is null, then assertion passes the test case, and the test case is marked as "passed", and if an object is not null, then assertion aborts the test case and the test case is marked as "failed". It aborts the test if object is null and gives an exception. **Syntax:**Assert.assertNull(object);	**assertNotNull:** The assertNotNull() method is used to check whether a particular object is NULL or not. This method throws an AssertionError if the object has some value (i.e., it is not NULL). In case of an assertion failure, the current test method is aborted with an exception. **Syntax:** Assert.assertNotNull(object);
String verifyAssertNull = null; assertNull(verifyAssertNull, "Expected value is not null");	driver.navigate().to("https://www.tronixtechs.com/"); String actualTitle = driver.getTitle(); assertNotNull(actualTitle, "Page title should not be null");

D. Assert Identical: assertSame, assertNotSame	
assertSame: The assertSame() method checks whether two objects refer to the same instance in memory. It tests whether the two objects are identical in terms of their memory address, rather than comparing their values. **Syntax:** assertSame(expected, actual);	**assertNotSame:** The assertNotSame() method asserts that two objects do not refer to the same object. If they do refer to the same object, an AssertionError without a message is thrown. **Syntax:** assertNotSame(expected, actual);

String str1 = "Selenium";
String str2 = "Selenium"; // Points to the same string pool object
Assert.assertSame("Objects are not identical!", str1, str2);
	String str1 = new String("Selenium");
	String str2 = new String("Selenium"); // Different objects in heap
	Assert.assertNotSame("Objects are identical!", str1, str2);

2. Soft Assertions: These types of Assertions are the type of assertions do not throw an exception when an assertion fails and continues with the next step after the assert statement. In a hard assertion, when the assertion fails, it terminates or aborts the test. If the tester does not want to terminate the script, they cannot use hard assertions. To overcome this, one can use soft assertions. Soft asserts log the failures and allow the test to continue executing, capturing all the failures encountered.

Common Soft Assertions:

softAssert.assertEquals(actual value, expected value)	Verifies that two values are equal
softAssert.assertNotEquals(actual value, expected value)	Verifies that two values are not equal
softAssert.assertTrue(Boolean Conditions)	Verifies that a condition is true
softAssert.assertFalse(Boolean Conditions)	Verifies that a condition is false
softAssert.assertNull(Object object)	Verifies that an object is null
softAssert.assertNotNull(Object object)	Verifies that an object is not null

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;
public class SoftAssertionExample {
@Test
public void testSoftAssertions() {
WebDriver driver = new ChromeDriver();    // Set up WebDriver
driver.get("https://www.example.com");
SoftAssert softAssert = new SoftAssert();   // Create SoftAssert object
```

```
String actualTitle = driver.getTitle();          // Soft assert for title
String expectedTitle = "Example Domain";
softAssert.assertEquals(actualTitle, expectedTitle, "Title does not match!");
String actualURL = driver.getCurrentUrl();       // Soft assert for URL
softAssert.assertNotEquals(actualURL, "https://wrong-url.com", "URL should not match
wrong URL!");
// Soft assert for boolean condition (check if title contains "Example")
softAssert.assertTrue(actualTitle.contains("Example"), "Title does not contain
'Example'!");
// Soft assert for false condition
softAssert.assertFalse(actualTitle.contains("Wrong"), "Title should not contain
'Wrong'!");
Object obj = null;   // Soft assert for null object
softAssert.assertNull(obj, "Object is not null!");
Object obj2 = new Object();      // Soft assert for not null object
softAssert.assertNotNull(obj2, "Object is null!");
System.out.println("Test execution continues even after assertion failures!");
// Report all assertion failures
softAssert.assertAll(); // If any assertion fails, it throws an exception here
driver.quit();      }   }
```

Difference Between Hard Asserts and Soft Asserts:

Hard Asserts	Soft Asserts
If you want your test case execution to proceed only after an assertion is passed (For Example, To Verify valid login and only then execute the other steps), then use Hard Assertions.	If you need to execute all the steps of a test case to be executed even after an assertion fails, and you also want to report assertion exception, then opt for using Soft Assertions.
An assertion error (java.lang.AssertionError) is thrown when the conditions in the Hard Assert are met.	The errors are accumulated in each @Test execution and the AssertAll() method throws asserts encountered during the process of Selenium Test Automation execution.
No need to invoke any method to view test results.	Tester needs to invoke assertAll(), to view assertions at the end of the test result.
Import the org.testng.Assert package. Assertion hardAssert = new Assertion();	Import the org.testng.asserts.SoftAssert package. Assertion softAssert = new SoftAssert();

****

WebElements in Selenium

Anything that is present on the web page is a WebElement such as text box, button, etc. WebElement represents an HTML element. Selenium WebDriver encapsulates a simple form element as an object of the WebElement. It basically represents a DOM [Document Object Model] element and all the HTML documents are made up by these HTML elements.

```
        html
        body
      My First Heading;
      My first paragraph;
   /body
/html
```

There are various techniques using which the WebDriver identifies the WebElements which are based on different properties like ID, Name, Class, XPath, Tagname, CSS Selectors, link Text, etc.

WebDriver provides two methods to find the elements on the web page.
findElement() – finds a single WebElement and returns it as a WebElement object.
findElements() – finds elements in a particular location using locators

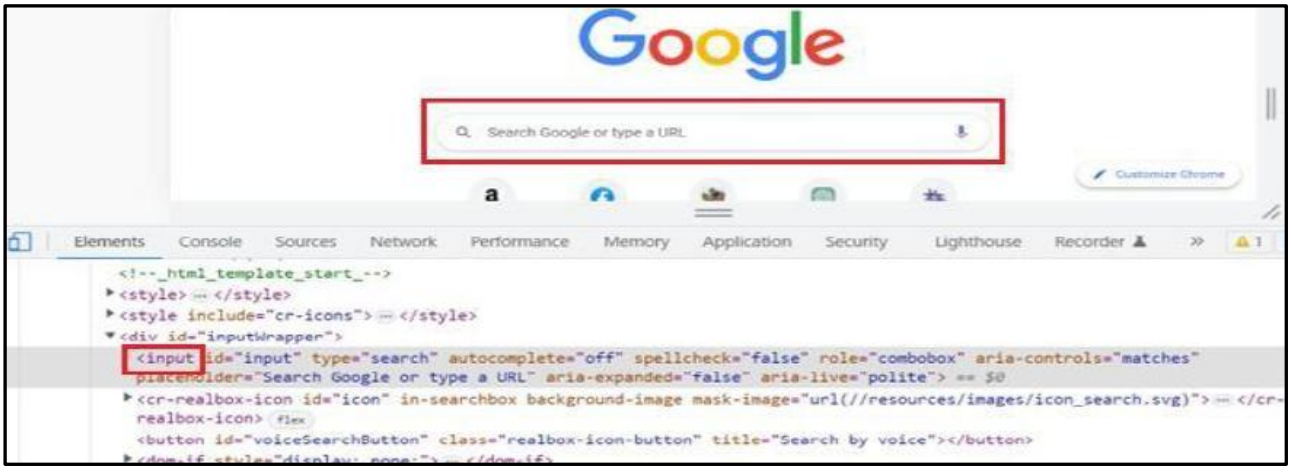
Types of Web elements in Selenium:

WebElements in Selenium can be divided into different types, namely:

1.	Text / Edit Box
2.	Text area
3.	Button
4.	Link
5.	Checkbox
6.	Radio button
7.	Image, Image Link, and Image Button
8.	Dropdown list

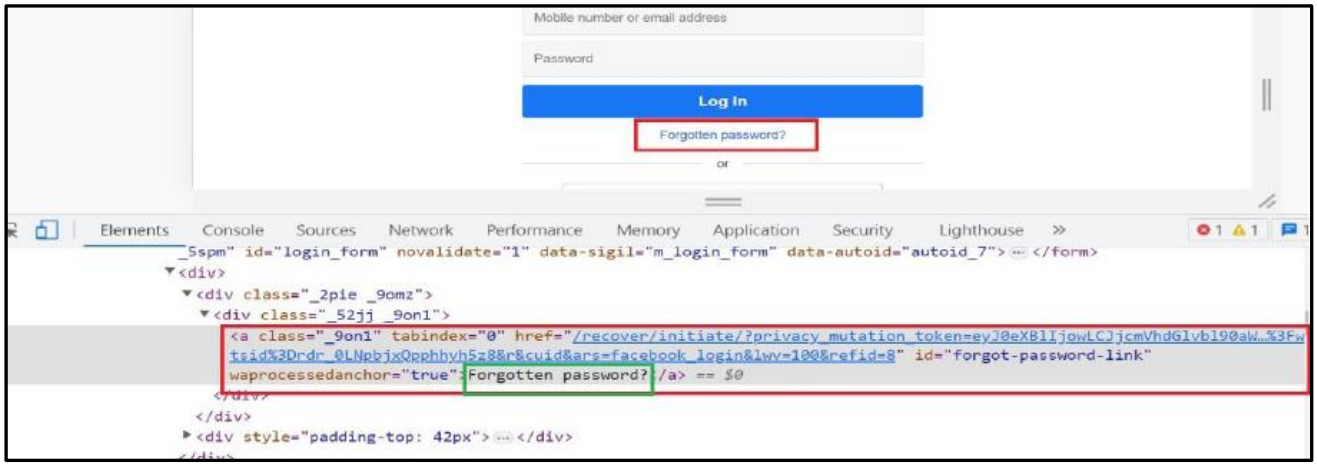
1. Text / Edit Box: It is a basic text control that enables a user to type a small amount of text. Selenium can be used to input text to an edit box. An edit box is represented by the input tag and its type attribute should have the value as text. It can be identified with any of the locators like - id, class, name, css, xpath and tagname. To input a value into an edit box, we have to use the method send_keys.

Ex: driver.findElement(By.xpath("//*[@name=\"q\"]")).sendKeys("Facebook");



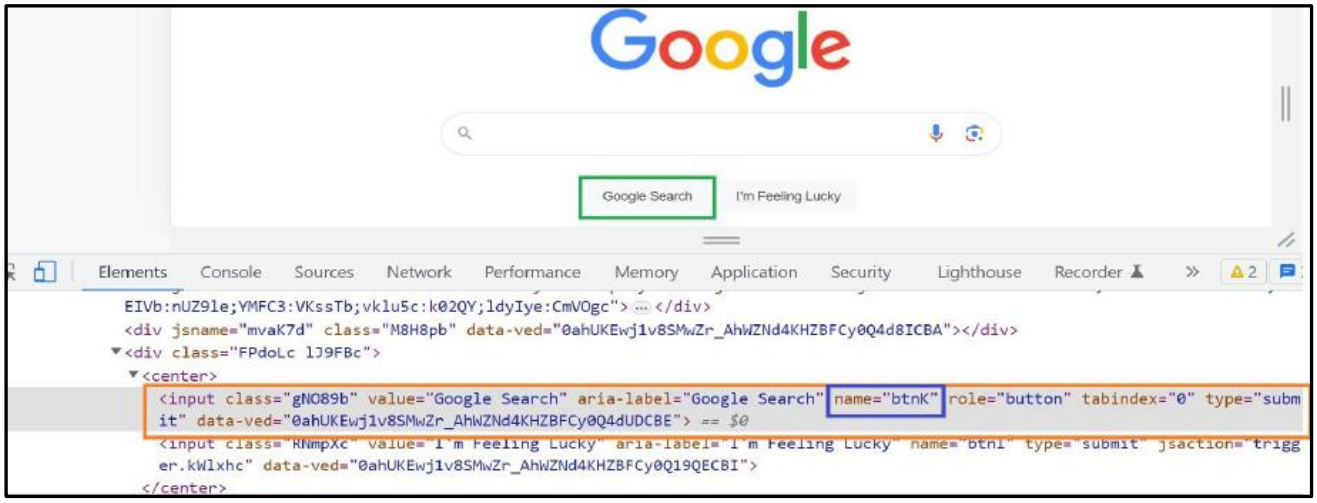
2. Link: link is more appropriately referred to as a hyperlink and connects one web page to another. It allows the user to click their way from page to page. A linkText is used to identify the hyperlinks on a web page. It can be determined with the help of an anchor tag (<a>). In order to create the hyperlinks on a web page, you can use anchor tags followed by the linkText. In general, there are three types of links: Internal links – hyperlinks that lead from one page to another within your own website; External links – hyperlinks that lead from your website to another resource; Backlinks – hyperlinks that lead from another site to yours. Links can be accessed using an exact or partial match of their link text.

Ex: driver.findElement(By.linkText("click here")).click();



3. Button: This represents a clickable button, which can be used in forms and places in the document that needs a simple, standard button functionality. A button is a widget used to perform actions such as submitting a form, opening a dialog, cancelling an action, or performing a command such as inserting a new record or displaying information. To click a button using Selenium in Java, find the button (web element) and then call click() function on this button web element. The button can be found by name, id, class name, etc.

Ex: driver.findElement(By.name("btnK")).click();



4. Image, Image Link, and Image Button:

Image: We can get the source of an image in Selenium. An image in an html document has <img> tagname. Each image also has an attribute src which contains the source of image in the page. To fetch any attribute in Selenium, we have to use the getAttribute() method.

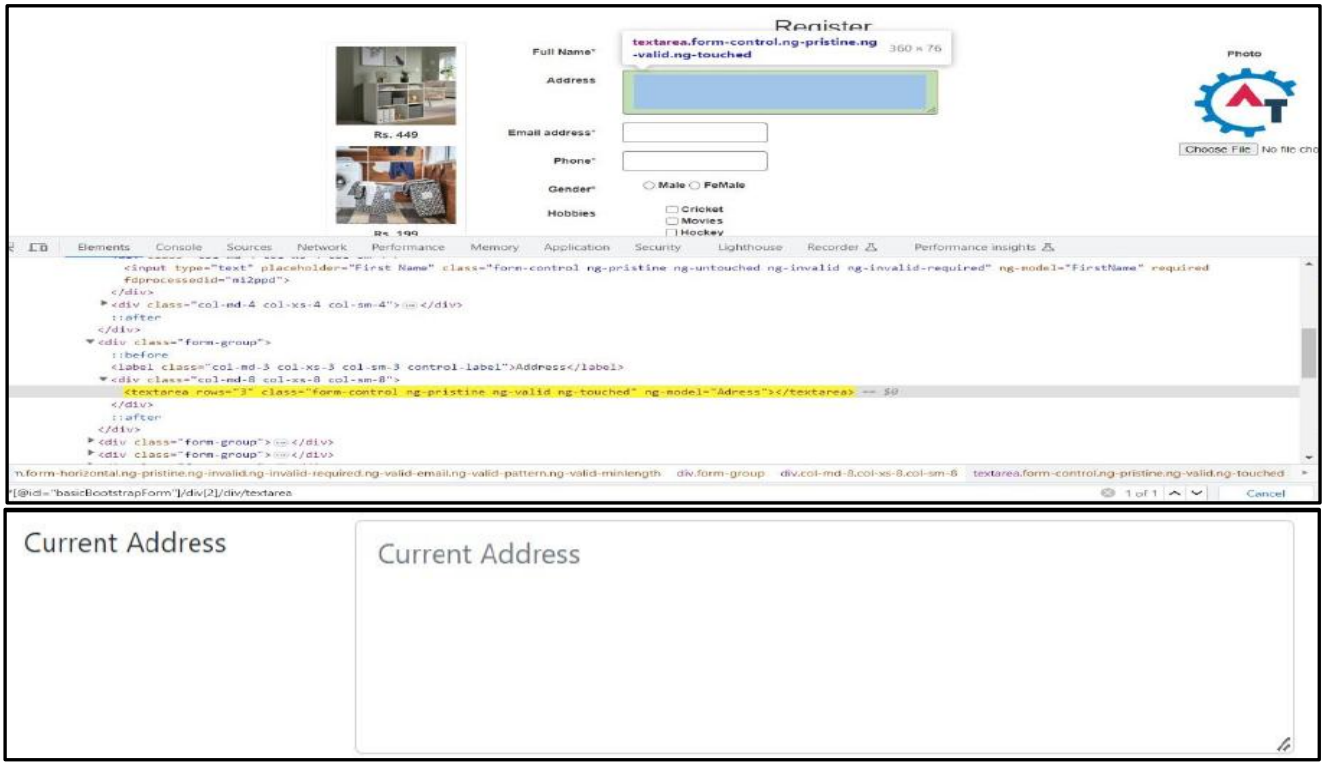
Image link: Image links in web pages are the links that represent an image. When we click an image, it navigates to a different window or a page. Since Image links have no link texts we cannot use the By.linkText() and By.partialLinkText() to access the image links. In this case, we use either By.cssSelector or By.Xpath.

Ex: driver.findElement (By.xpath("//img[@src="https://static.xx.fbcdn.net/rsrc.php/y8/r/dF5SId3UHWd.svg"]")). click();

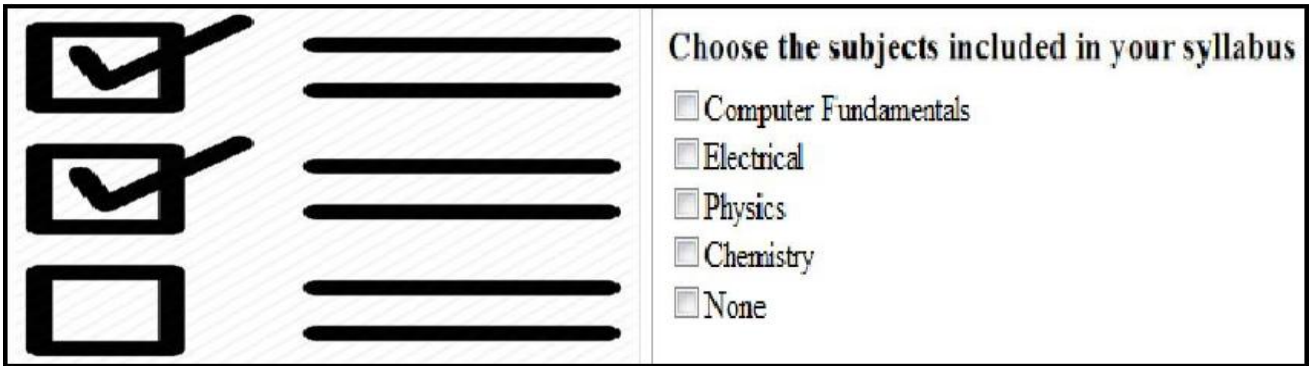
5. Text Area: A textarea is designed to input more than one line of text into a form. The size of the textarea can be specified in two dimensions, allowing for a much larger input space than a textbox. A TextArea object is a multi-line region that displays text. It can be set to allow editing or to be read-only. TextField helps to enter a single line of text.

TextArea helps to enter multiple lines of text. A text area is identified with textarea tagname in the html code. Let us input some text inside the below text area, then clear the text.

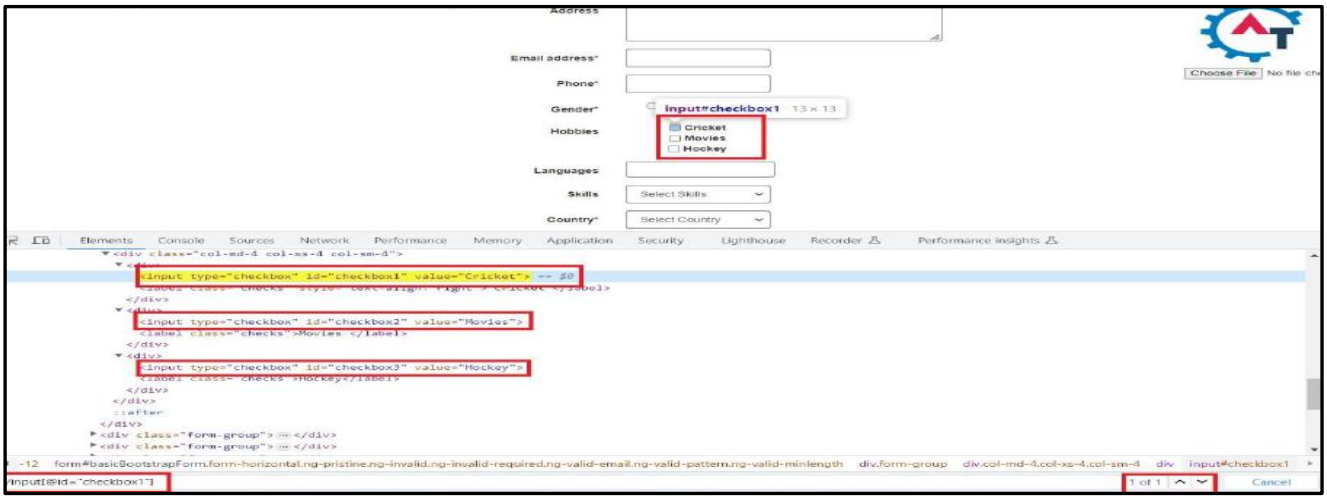
The <textarea> tag defines a multi-line text input control. The <textarea> element is often used in a form, to collect user inputs like comments or reviews. A text area can hold an unlimited number of characters, and the text renders in a fixed-width font (usually Courier).
Ex:driver.findElement(By.xpath("//*[@id="basicBootstrapForm"]/div[2]/div/textarea")).click ();



6. CheckBox: The checkbox is a Web element that is used for the selection of one or multiple options. The checkbox can be turned on or off (that is checked or unchecked). A checked Checkbox is the one that is seen as the tick marked. Unchecked is the one that is unticked. Checkboxes are widely used on web pages to offer multiple choices of an option to the user. Users may get a list of choices, and the checkbox records the choices made by the user. The checkbox allows users to select either single or multiple choices out of the given list. In Selenium, a checkbox refers to an HTML input element of type "checkbox". We can define a checkbox in HTML using <input type="checkbox"> tag. It is a form element that allows users to select or deselect a particular option. A CheckBox on an HTML page provides various unique properties that can identify and automate their behavior in Selenium WebDriver. Checkboxes are commonly used in web forms to provide multiple selection options.



In Selenium WebDriver, you can interact with checkboxes using various methods provided by the WebElement class. A check box on/off is also done using the click() method.



Selenium Provides Methods to handle Checkboxes such as:

- ✓ isSelected(): This method is used to determine if the checkbox is currently selected or not. It returns a boolean value (true if selected, false otherwise).
- ✓ click(): This method is used to click on the checkbox, toggling its selected state. If the checkbox is selected, it will be deselected, and if it is deselected, it will be selected.

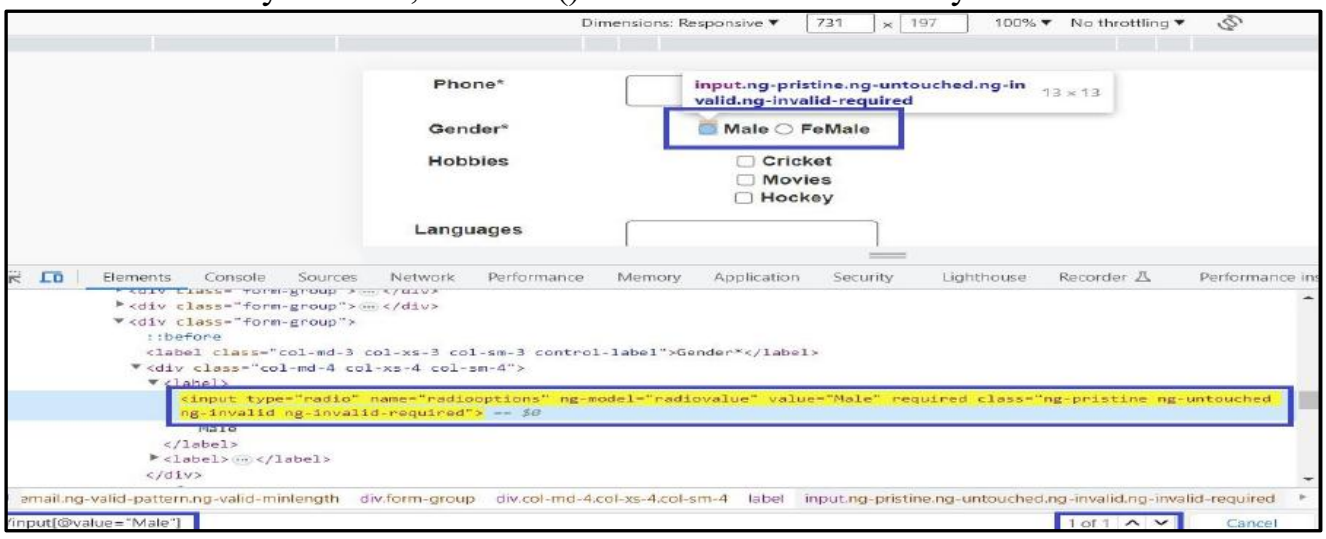
7. Radio Button: A Radio button is nothing but a graphical element that controls the selection of options, thereby allowing the user to select any one option from a set of options. It is also known as the options button. Radio buttons are so-called because of the functionality they have like channels of Radio i.e. only one button can be selected at a time. Radio buttons are commonly used in web forms, surveys, and other scenarios where only one option can be selected at a time. When a radio button is selected, it displays a dot or a filled-in circle indicating the chosen option. Selecting one radio button automatically deselects any other radio buttons within the same group. In Selenium, a radio button is a type of input element that allows users to select a single option from a group of mutually exclusive choices. It is represented by the HTML <input> tag with the type="radio" attribute.

Fig: An example of Radio Buttons

To interact with radio buttons using Selenium WebDriver, you can use the WebElement class's methods to perform actions such as selecting a radio button, verifying its selection state, or retrieving its value.

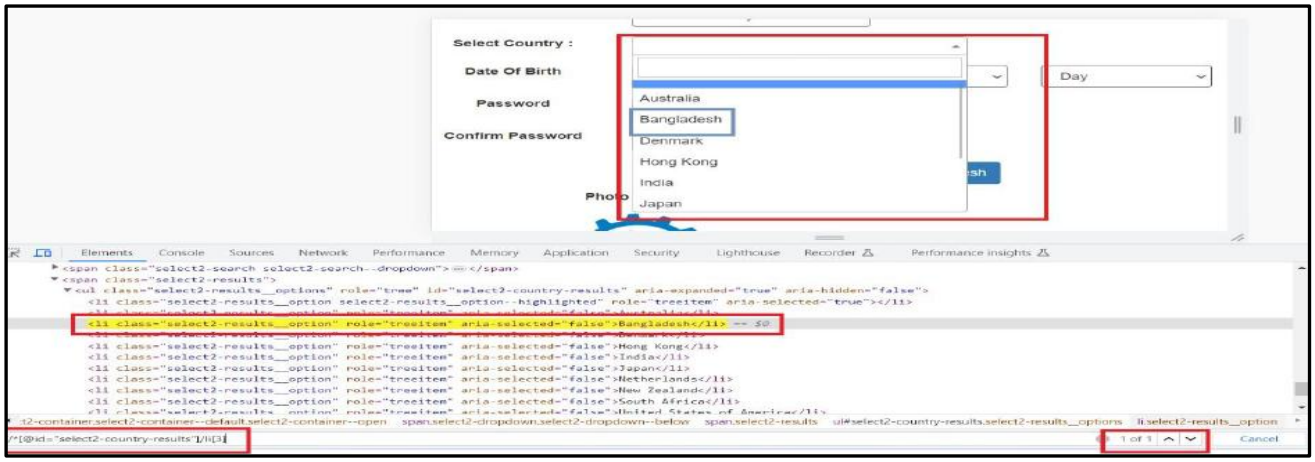
Here are some commonly used methods to work with radio buttons in Selenium:

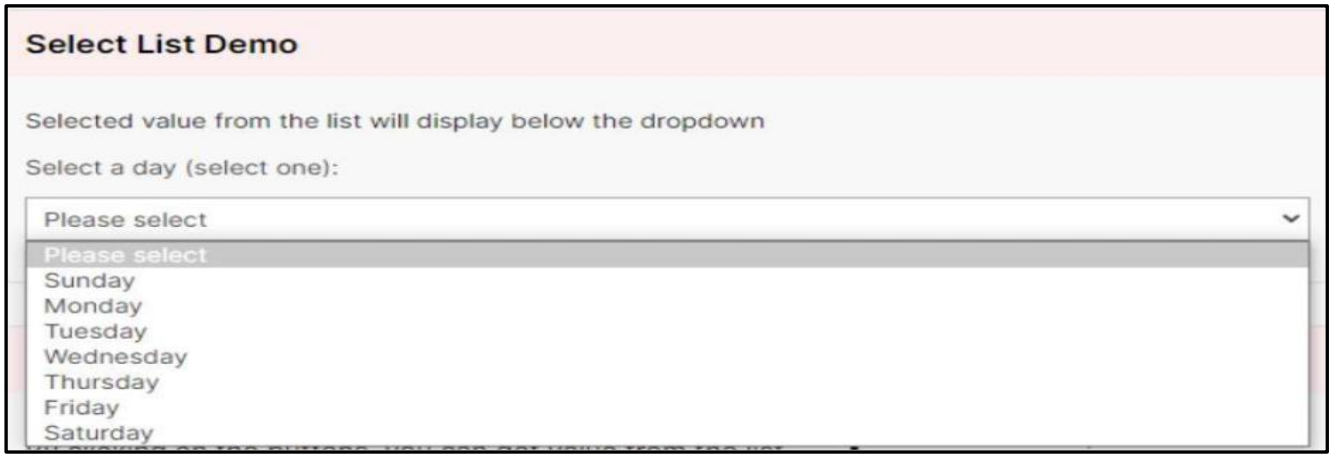
- ✓ **isSelected():** This method is used to check if a radio button is selected or not. It returns a boolean value indicating the radio button's current selection state.
- ✓ **click ():** This method simulates a mouse click on the radio button, selecting it. If the radio button is already selected, the click() method will not have any effect.



8. Dropdown List: A dropdown field is a form element that presents a list of options to a user as a dropdown menu. The user can select a single option from the menu by clicking on it. The selected option is then displayed in the dropdown field. Dropdown fields are often used in forms to allow users to select a value from a predefined list of options, such as a list of countries or a list of product categories or picking a date from a calendar, or choosing a category from a list of options. They can be created using HTML and styled using CSS. In Selenium, a "drop-down" generally refers to a web element that presents a list of options when clicked or activated. These options typically appear in a menu that "drops down" from the original element, hence the name "drop-down." To select drop down options via automation, Selenium is one of most widely used automation libraries to automate websites. Drop downs in a website could be created in several different ways. Some dropdowns are created using <select> HTML tag and some others are created using <ul> ,<li>, <button> and tags.

Selenium WebDriver provides Select class which can be used only for drop down created using <select> HTML tag. Select class has methods such as selectByVisibleText(), selectByIndex() and selectByValue() to select the desired option from dropdown.





Select Class in Selenium:

The Select Class in Selenium is a method used to implement the HTML SELECT tag. The html select tag provides helper methods to select and deselect the elements. The Select class is an ordinary class so new keyword is used to create its object and it specifies the web element location.

Select Methods in Selenium: The following are the most common methods used on Selenium dropdown list.

a. selectByVisibleText() and deselectByVisibleText(): Selects/deselects the option that displays the text matching the parameter.

Parameter: The exactly displayed text of a particular option

Example: drpCountry.selectByVisibleText("ZIMBABWE");

```
Select drpCountry = new Select(driver.findElement(By.name("country")
drpCountry.selectByVisibleText("ZIMBABWE");
```

b. selectByValue() and deselectByValue(): Selects/deselects the option whose "value" attribute matches the specified parameter. Remember that not all dropdown options have the same text and "value", like in the example below.

Parameter: Value of the "value" attribute

```
<option value="10">ANGUILLA </option>
<option value="234">ANTARCTICA </option>
<option value="1">ANTIGUA AND BARBUDA </opti
```

Example: drpCountry.selectByValue("234");
select.selectByValue("search-alias=mobile-apps");

c. selectByIndex() and deselectByIndex(): Selects/deselects the option at the given index.

Parameter: The index of the option to be selected.

Example: drpCountry.selectByIndex(0);
select.select_by_index(3); [# Selects the fourth option (index 3)]

d. deselectAll (): Clears all selected entries. This is only valid when the drop-down element supports multiple selections.

Parameter: Not needed

Example: drpCountry.deselectAll();

****

Locaters In Selenium

Selenium WebDriver- Locating Strategies:

Selenium uses a concept of 'Locators' for identifying objects. The findElement method in Selenium can be used to identify the elements. It takes a locator or a query object 'By' object and returns an object of type WebElement. 'By' object can be used with the various locator strategies available. Finding an element and confirming the expected result requires locators. Through locator, In Document Object Mode (DOM) i.e. webpage, we uniquely identify the object. The 'By' class is used in WebDriver to locate the elements. Locators Used In Selenium For Object Identification.

"Identify the objects such as Links, Buttons, Edit boxes, Drop downs, etc on the application Selenium uses a concept called "Locators"".

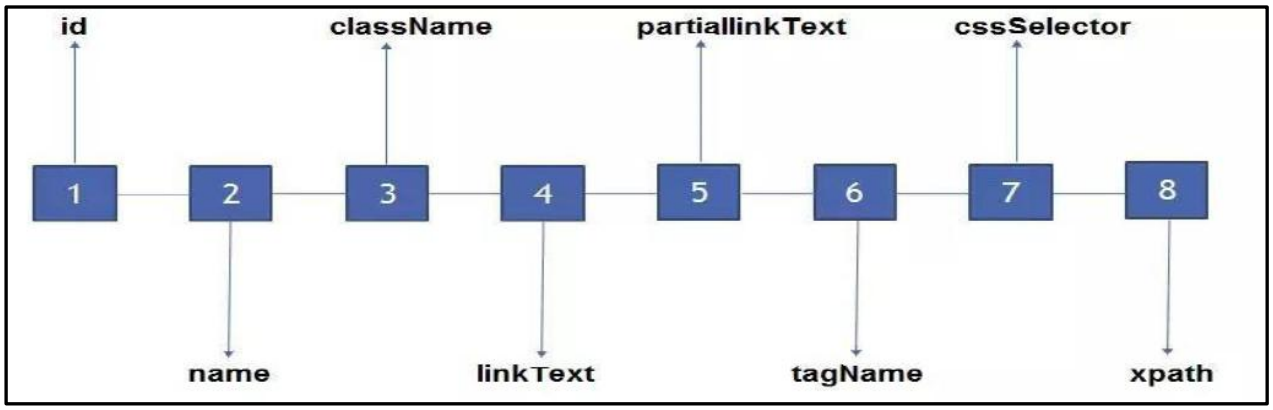
Locating elements in Selenium WebDriver is performed with the help of findElement() and findElements() methods provided by WebDriver and WebElement class.

findElement() returns a WebElement object based on a specified search criteria or ends up throwing an exception if it does not find any element matching the search criteria.

findElements() returns a list of WebElements matching the search criteria. If no elements are found, it returns an empty list.

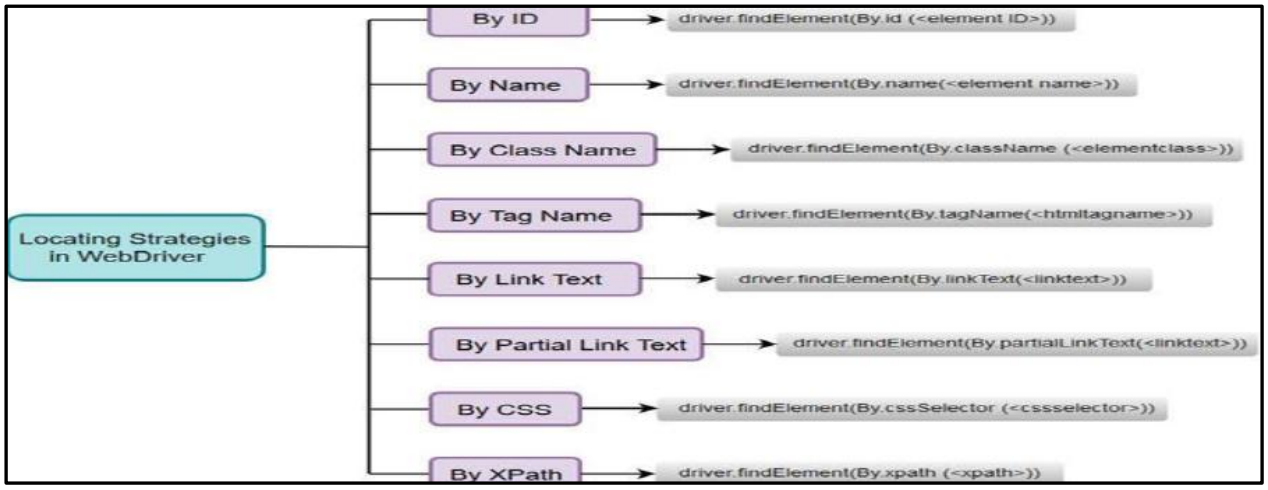
Selenium provides us 8 different types of locators or Identifiers to identify the elements on a webpage, which are as follows:

1.	Id.
2.	Name.
3.	Class Name.
4.	Tag Name.
5.	Link Text.
6.	Partial Link Text.
7.	CSS.
8.	XPath.



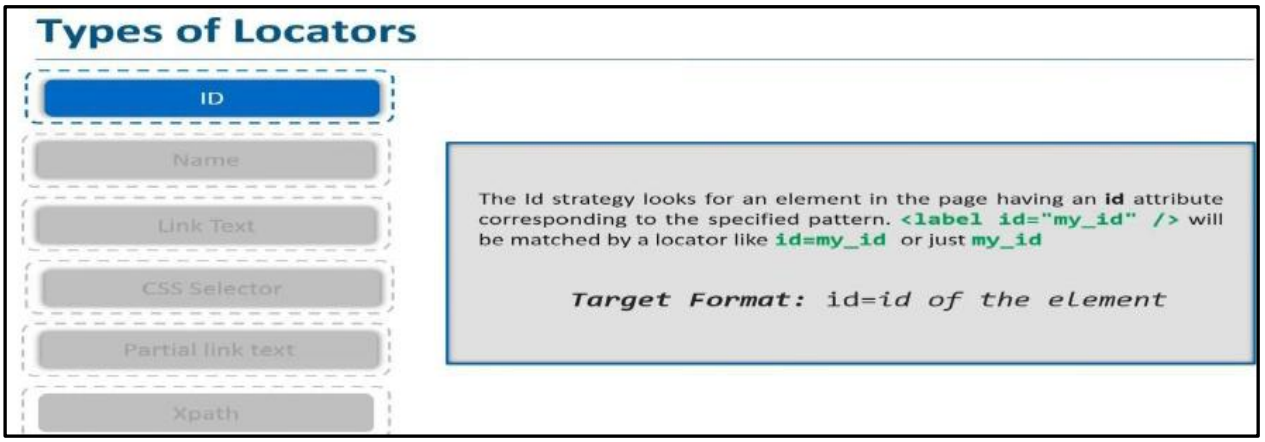
Just like Selenium IDE, WebDriver uses the same set of locating strategies for specifying location of a particular web element. Since, we are using WebDriver with java; each locating strategy has its own command in Java to locate the web elements.

A list of Locating Strategies used in WebDriver:



1. ID: The most efficient and preferred way to locate web elements on a web page is by ID as ID will be unique for every element. Thus, IDs being the safest and fastest option to identify an element is the first choice for object identification. Since ID's are supposed to be unique for each element. In Selenium, the ID locator is used to locate elements based on their unique id attribute. The id attribute is an HTML attribute that provides a unique identifier for an element on a webpage. It is commonly used to uniquely identify specific elements. To locate an element using the ID locator in Selenium with Java, you can use the By.id() method.

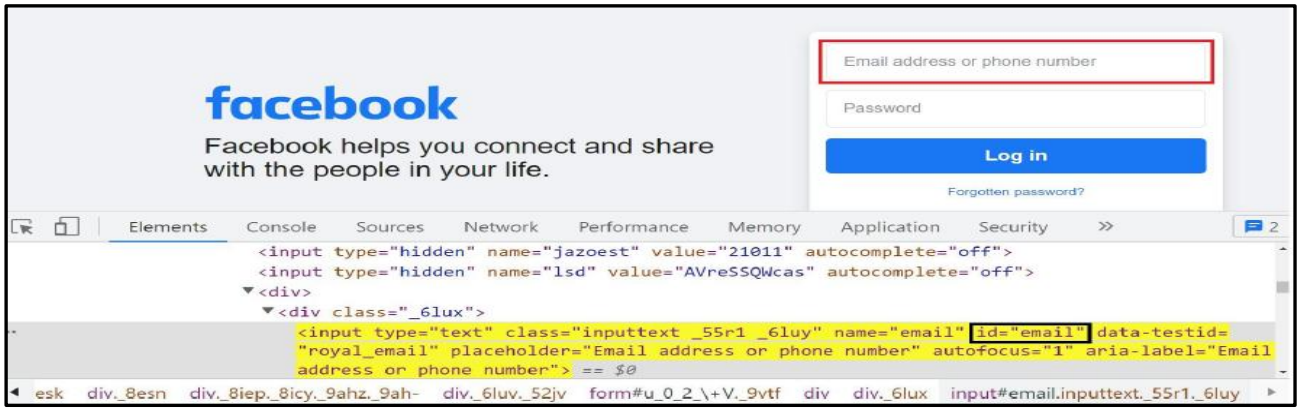
But some cases IDs may not be mentioned in HTML codes of the webpage. In such cases you would have to go for other locators.



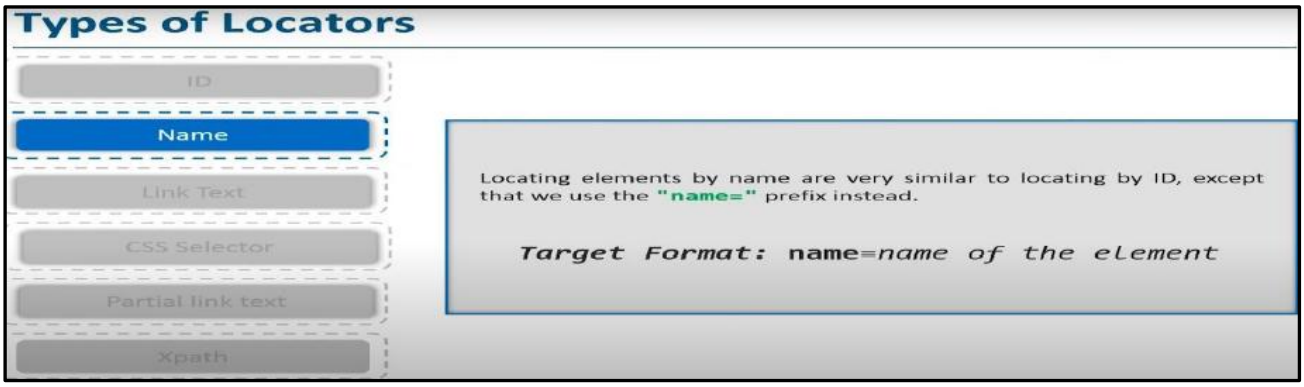
For this example, we will use Facebook as our test app.

Use this demo page https://www.facebook.com/ for testing. Inspect the "Email address or phone number" text box using its ID locater. In this case, the ID is "**email**".

Syntax: driver.findElement(By.id("Element ID"));
Ex: driver.findElement(By.id("email "));



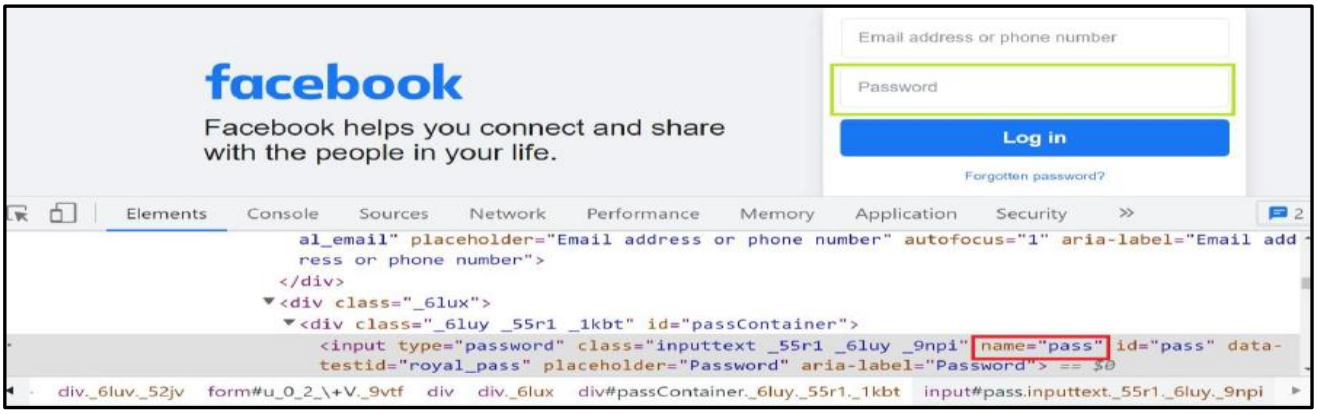
2. Name: The second preferred choice if ID is not mentioned for the element. But ensure the Name attribute is unique in the webpage because like IDs, Name attribute need not be unique always. So, there is a chance that there may be other elements having same Name and your script might end up selecting the wrong element. Locating elements by name are very similar to locating by ID, except that we use the "name=" prefix instead. In Selenium, the Name locator is used to locate elements based on their name attribute. The name attribute is an HTML attribute that is commonly used to provide a name for an element. It may or may not be unique on a webpage. To locate an element using the Name locator in Selenium with Java, you can use the By.name() method.



Here's an example:

Syntax:	driver.findElement(By.name("Element Name"));
Ex:	driver.findElement(By.name("pass")).sendKeys("**********");

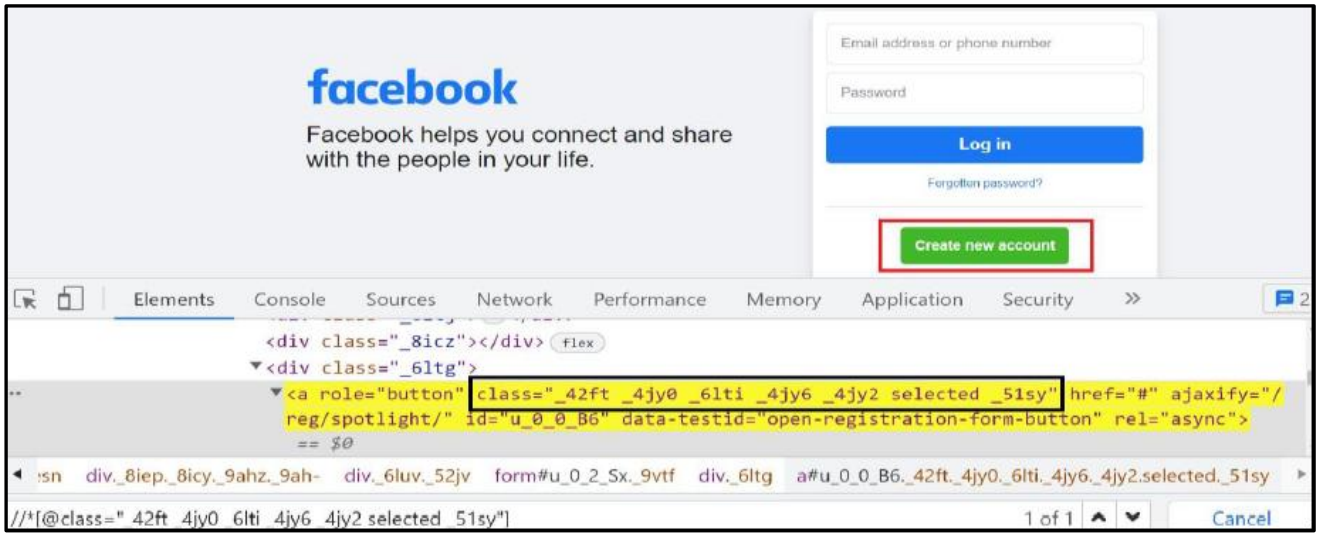
Here an object is accessed with the help of names. In this case, it is the name of the text box. Values are entered into the text box using the sendkeys method with the help of NAME(pass).



3. Class Name: In Selenium, the By.className() method is used to locate elements based on their class name. The class name is one of the attributes of an HTML element that can be used for identification. This is yet another locator used for identifying web elements. But if you are using Class name for element identification, make sure it is unique. There may be many elements with same class name. This locator is often used in identifying a group of elements having same class and performing some operations on it. Here an object is accessed with the help of Class Names. In this case, it is the Class name of the WebElement. The Value can be accessed with the help of the gettext method. The class name is stored in the class attribute of an HTML tag.

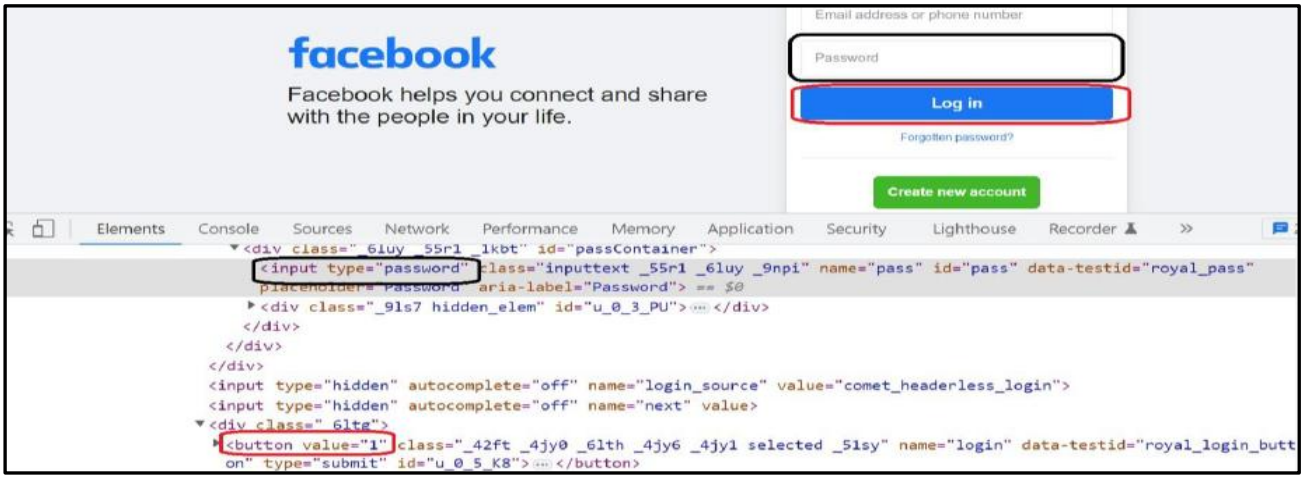
In-order to use Class name locator in Selenium, we need to use the below syntax:

Syntax: driver.findElement(By.className("Element className"));
Ex: driver.findElement(By.className("_42ft _4jy0 _6lti _4jy6 _4jy2 selected _51sy"));



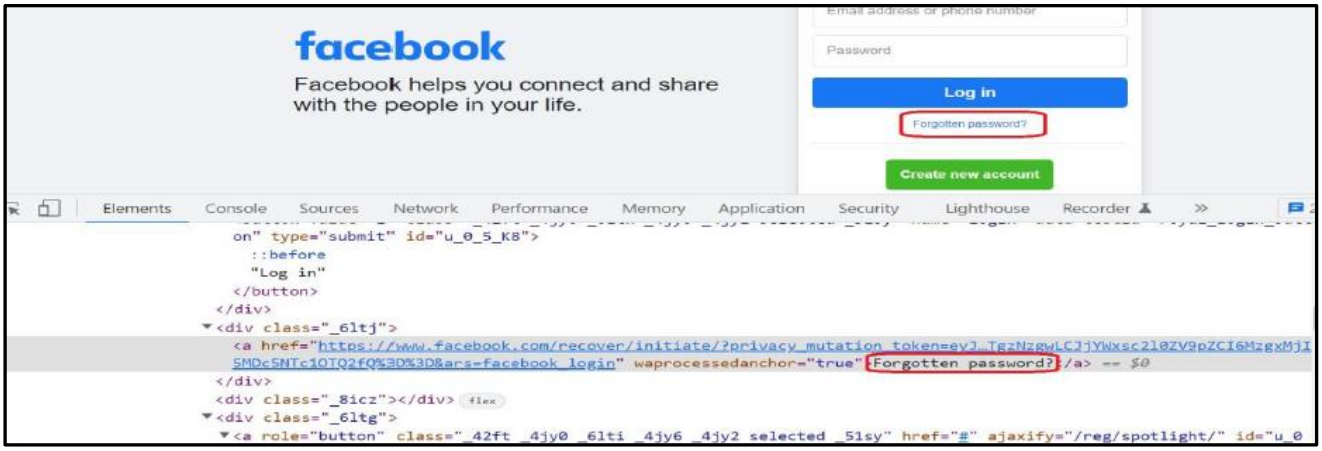
4. TagName: Tag Name locator is used to find the elements matching the specified Tag Name. It is very helpful when we want to extract the content within a Tag. Here Selenium finds the elements using tag name. Tag names are a good choice for group elements like Select/Check boxes, drop down menu, etc. A tag name is a part of a DOM structure where every element on a page is been defined via tag like input tag, button tag or anchor tag etc. Each tag has multiple attributes like ID, name, value class etc. Tag Name of an element can be used to locate that particular element in the WebDriver. It is very easy to handle tables with the help of this method.

Syntax: driver.findElement(By.tagName("Element TagName"));
Ex: driver.findElement(By.tagName("input"));

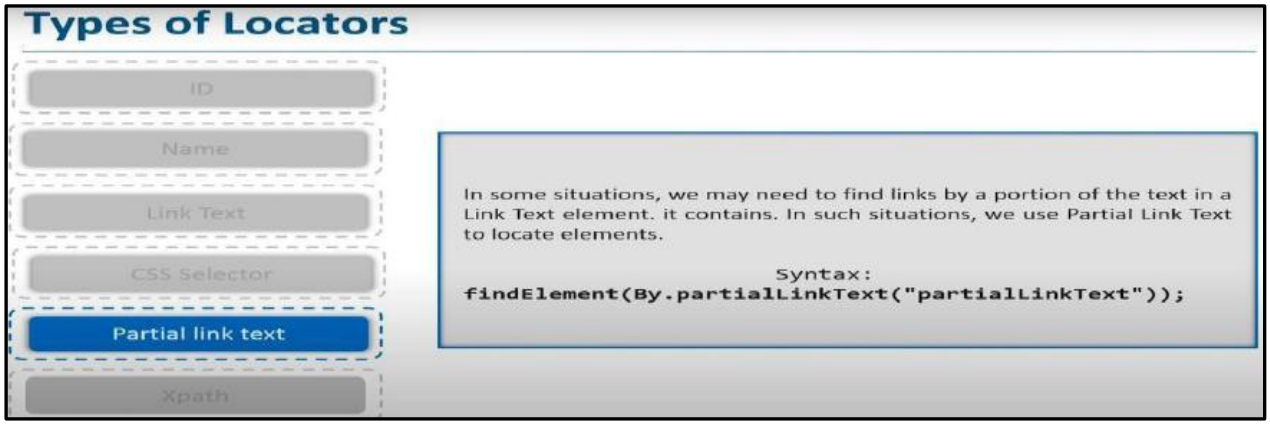


5. Link Text: A linkText is used to identify the hyperlinks on a web page. It can be determined with the help of an anchor tag (<a>). The link text is the visible clickable text in a hyperlink. It is the text in between the anchor tags. Here also, make sure of uniqueness. There may be multiple links with same text such as in repeated header and footer menus. If there are repeated occurrences, Selenium will perform the action on the first matching element with the link text.You can identify the hyperlinks on a web page using this linkText. To create the hyperlinks on a web page, you can use the anchor tags followed by the link text. Accessing links using their exact link text is done through the By.linkText() method. However, if there are two links that have the very same link text, this method will only access the first one. Use the LinkText locator strategy in Selenium to locate elements based on the text of a link.

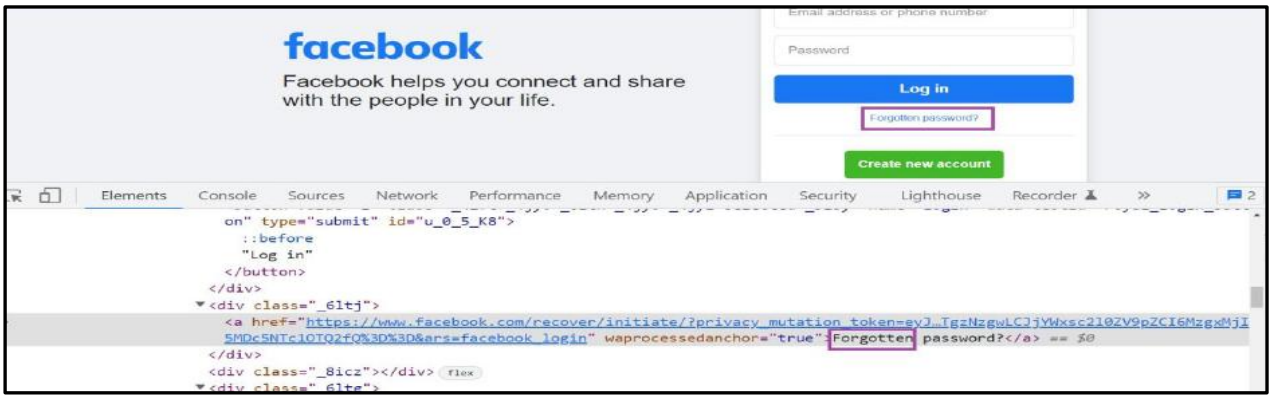
Syntax:	driver.findElement(By.linkText("Link Text"));
Ex:	driver.findElement(By.linkText("Forgotten password?"));



6. Partial LinkText: You can use partial LinkTexts also same way as you use LinkTexts. This is very helpful in cases where you know part of the link text would never change in the application. So that even if remaining part of link text changes your code doesn't break. In some cases, you may need to find links by a portion of the text in a linkText element. In such situations, you can use Partial Link Text to locate elements. The partial link text locator matches the text inside the anchor tag partially. Accessing links using a portion of their link text is done using the By.partialLinkText() method. If you specify a partial link text that has multiple matches, only the first match will be accessed.



Syntax:	driver.findElement(By.PartialLinkText("PartialLink Text"));
Ex:	driver.findElement(By.PartialLinkText("Forgotten"));



7. CSS Selectors: CSS Selectors are used within Selenium to locate elements on a web page. They are beneficial because they are shorter than XPath locators and allow for a more clear and crisp method to locate the element. They work by using patterns to match tags and their attributes, such as class or ID. CSS is used for providing styles to your webpage. These can be used for identifying elements in the web page. This is considered to be a fastest option for identifying elements. Locating elements by CSS selectors is the preferred way as it is faster and more readable than XPath.CSS Selectors in Selenium are string patterns used to identify an element based on a combination of HTML tag, id, class, and attributes. Locating by CSS Selectors in Selenium is more complicated than the previous methods, but it is the most common locating strategy of advanced Selenium users because it can access even those elements that have no ID or name. CSS expression also has a syntax similar to XPath.CSS selector is a powerful locator strategy that allows you to find elements on a web page based

on CSS selectors. CSS selectors offer a wide range of options to locate elements based on attributes, classes, IDs, element types, and more.

How to create a CSS Selector?

CSS Selector syntax is quite similar to the XPath syntax. It can be represented syntactically as follows:

CSS expression= Node/ Tagname [attribute_name = 'attribute_value']

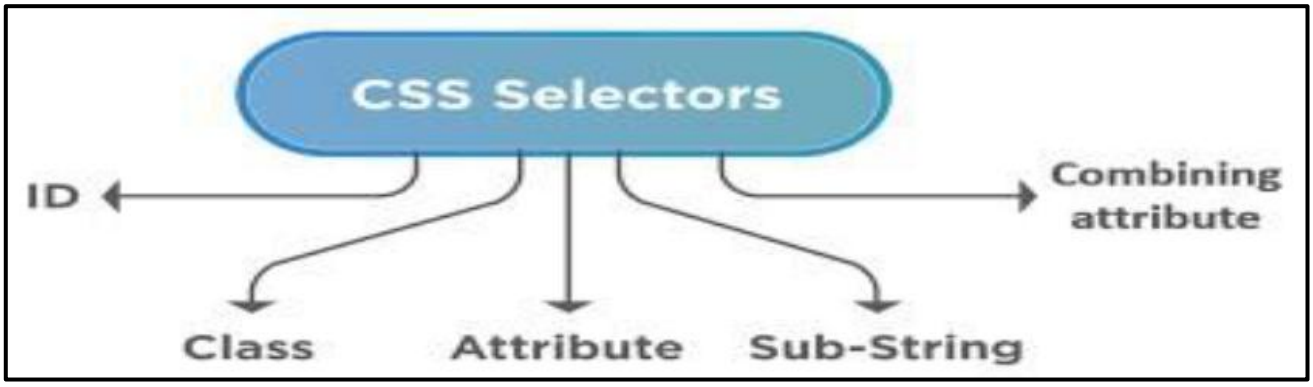
Where,

- **Node/ Tagname** is the tag name of the HTML element, which needs to locate.
- **attribute_name** is the name of the attribute which can locate the element.
- **attribute_value** is the value of the attribute, which can locate the element.

To identify an element using CSS in Selenium:

Syntax:	driver.findElement(By.cssselector("Element css expression"));
Ex:	driver.findElement(By.cssSelector("input#email"));

Types of CSS Selectors (with Examples): There are 5 types of CSS Selectors.



A. CSS Selector – ID: The id selector uses the id attribute of an HTML element to select a specific element. The id of an element is unique within a page, so the id selector is used to select one unique element! To select an element with a specific id, write a hash (#) character, followed by the id of the element.

In this example, the script will access the "Email address or phone number" text box on the login form at facebook.com. The text box carries an ID attribute with the value "Email address or phone number". In this case, the ID attribute and its value are utilized to create a CSS selector that allows access to the text box. In CSS, we can use # notation to select the id attribute of an element.

How to create the Selenium CSS Selector for the web element

Locate the web element – “Email address or phone number” textbox. The HTML tag here is “input” and the ID attribute’s value is “Email address or phone number”. Combined, they refer to the “Email address or phone number” textbox. This is the data required to create the CSS selector.

Example: driver.findElement(By.cssSelector("input#email"));

Or) driver.findElement(By.cssSelector("#email"));

B. CSS Selector- Class: CSS Selector in Selenium using an HTML tag and a class name is similar to using a tag and ID, but in this case, a dot (.) is used instead of a hash sign. The class selector selects HTML elements with a specific class attribute. To select elements with a specific class, write a period (.) character, followed by the class name.

Syntax: css=<HTML tag><.><Value of class attribute>

"." is used to symbolize Class attribute. It is mandatory to use dot sign if a class attribute is being used to create CSS Selector.

The value of class is always preceded by a dot sign.

In this example, the script will access the “Email address or phone number” text box on the login form at facebook.com. The text box carries an ID attribute with the value “Email address or phone number”. In this case, the ID attribute and its value are utilized to create a CSS selector that allows access to the text box. In CSS, we can use # notation to select the id attribute of an element.

How to create the Selenium CSS Selector for the web element

Locate the web element – “Email address or phone number” textbox. The HTML tag here is “input” and the ID attribute’s value is “Email address or phone number”. Combined, they refer to the “Email address or phone number” textbox. This is the data required to create the CSS selector.

Example: driver.findElement(By.cssSelector("input.email"));

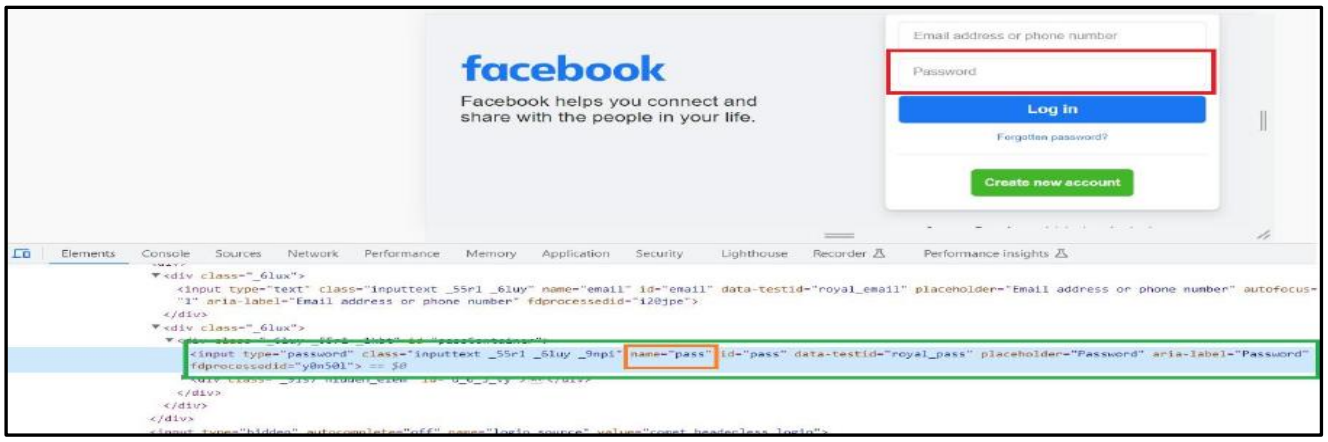
Or) driver.findElement(By.cssSelector(".email"));

C. CSS Selector – Attribute: CSS attribute selectors are used select and style HTML elements with the specified attributes and values. The [attribute] selector selects all the elements that have the specified attribute. The [attribute="value"] selector selects all the elements with the specified attribute and value.

Syntax: css=<HTML tag><[attribute=Value of attribute]>
css=tag[attribute=value]

Ex: input[name='pass']
driver.findElement(By.cssSelector("input[name='pass'"")).sendKeys("555");

We can use value, type and name as an Attribute to create a CSS Selector. Value of attribute denotes the value which is being accessed at the time of using a particular attribute.



D. CSS Selector: Sub-string: It allows matching a partial string to locate a particular web element. In CSS, a substring can be selected within an attribute value using the substring matching attribute selectors.

A Sub-string contains three methods for identifying the web element uniquely.

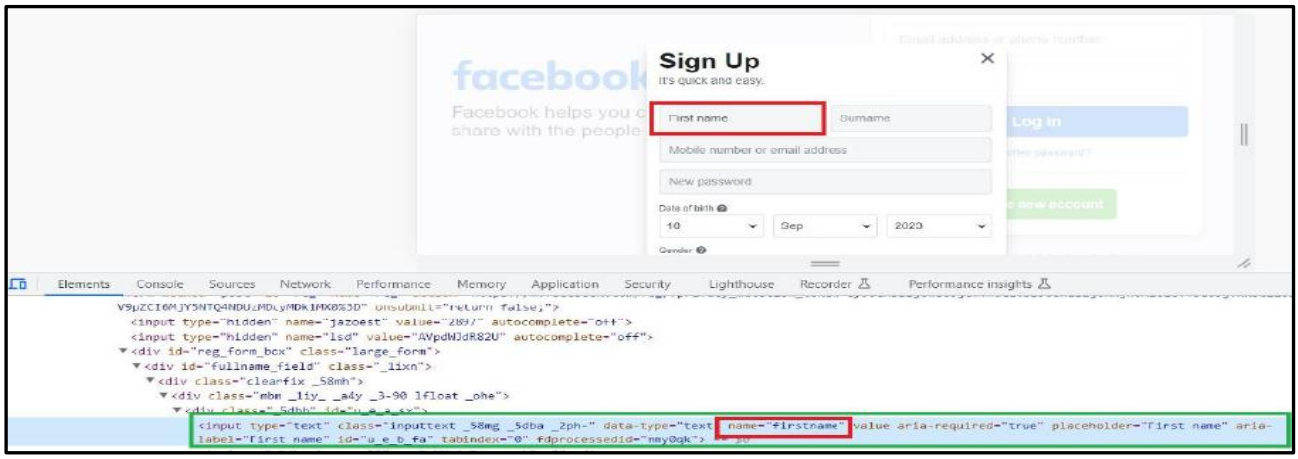
The sub-string methods are as follows:

i. **Contains (*):** The Contains method is used to identify the web element using * (i.e., asterisk) symbol.

Syntax: Tagname[attribute*='partial Attribute value'] Where,

[*]=the asterisk symbol is used to identify that the particular attribute contains any value or not. *=: The *= selector matches an element if the specified substring appears anywhere within the attribute value.

Example: input[name*='firstn'] Note: FB Login Page example
selects all <input> elements with a name attribute containing the substring " firstname ".



ii. Starts with (^): The Starts-with is used to identify the web element using ^ (i.e., carat) symbol.

Syntax of starts-with: Tagname[attribute^ =’Attribute starting value’] Where,
[^]=the carat symbol is used to identify the web element with its starting value.

Ex: input[name^='firstname']

^= : The ^= selector matches an element if the attribute value starts with the specified substring. Example: a[href^="https://"]

selects all <a> elements with an href attribute that starts with "https://".

iii. Ends with ($): The ends-with is used to identify the web element using $ (i.e., dollar) symbol.

Syntax for ends with: Tagname[attribute$ =’value’]

[$]=the dollar symbol is used to identify the web element with its ending value.

$=: The $= selector matches an element if the attribute value ends with the specified substring.

Example: input[placeholder$="First name"] img[src$=".png"] selects all <img> elements with a src attribute that ends with ".png".

All Locators Table:

Method	Syntax	Description
By.id	driver.findElement(By.id(<element ID>))	Locates an element using the ID attribute
By.name	driver.findElement(By.name(<element name>))	Locates an element using the Name attribute
By.className	driver.findElement(By.className (<element class>))	Locates an element using the Class attribute
By.tagName	driver.findElement(By.tagName (<htmltagname>))	Locates an element using the HTML tag
By.linkText	driver.findElement(By.linkText (<linktext>))	Locates a link using link text
By.partialLinkText	driver.findElement(By.partialLinkText (<linktext>))	Locates a link using the link's partial text
By. cssSelector	driver.findElement(By.cssSelector(<css selector>))	Locates an element using the CSS selector
By.xpath	driver.findElement(By.xpath (<xpath>))	Locates an element using XPath query

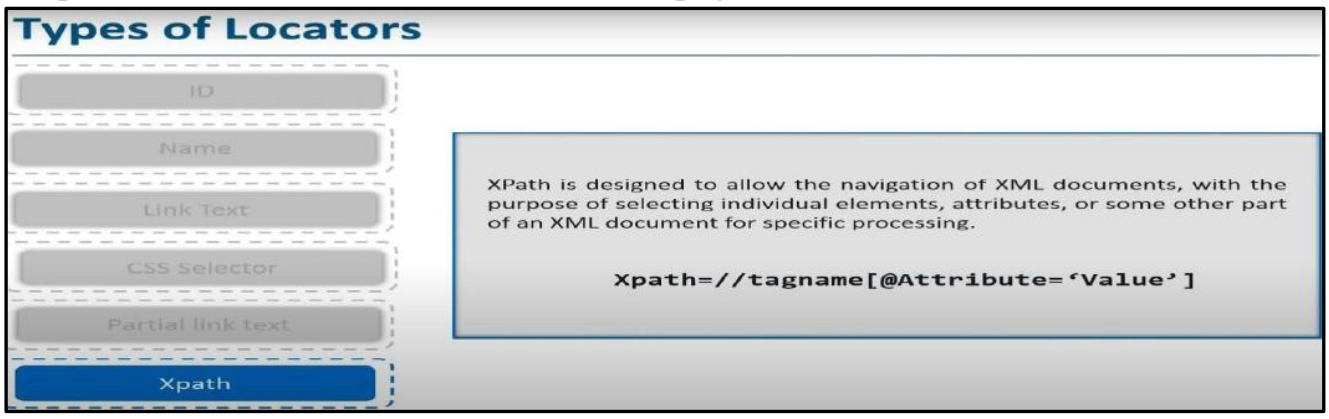
Locator	Syntax
className	driver.findElement(By.className (“value”))
cssSelector	driver.findElement(By.cssSelector(“value”))
id	driver.findElement(By.id(“value”))
linkText	driver.findElement(By.linkText(“value”))
name	driver.findElement(By.name(“value”))
partialLinkText	driver.findElement(By.partialLinkText(“value”))
tagName	driver.findElement(By.tagName (“value”))
xpath	driver.findElement(By.xpath(“value”))

8. Xpath: XPath is a Selenium technique that is used to navigate through the HTML structure of a webpage. It is a syntax or language that makes finding elements on a webpage possible using XML path expression. The XML Path Language (XPath) is used to uniquely identify or address parts of an XML document. In Selenium automation, there may be times when elements cannot be found with general locators like ID, NAME, CLASS, etc. And this is when XPath is used to locate those elements on the webpage. XPath in Selenium may be used on both XML and HTML documents. XPath is also known as XML Path.

XPath is a Selenium technique that is used to navigate through the HTML structure of a webpage. It is a syntax or language that makes finding elements on a webpage possible using XML path expression. It enables testers to navigate any document's XML structure, which can be used on both HTML and XML documents. While other locators in Selenium that search for elements using tags or CSS class names are more straightforward, they may not be sufficient to select all **DOM** elements of an HTML document. XPath is the used for navigation through

XML documents. Since an HTML document is also an XML document, every element in a web page can be identified using XPath. XPath is a Selenium technique to navigate through a page's HTML structure. XPath is one of the most commonly used locators in Selenium WebDriver that can help you navigate through the HTML structure of a page.

Note: DOM [Document Object Model] in Selenium is a programming interface that represents the structure of an HTML or XML document. It provides a way to access and manipulate the elements and content of a web page.



How to write XPath in Selenium using Java?

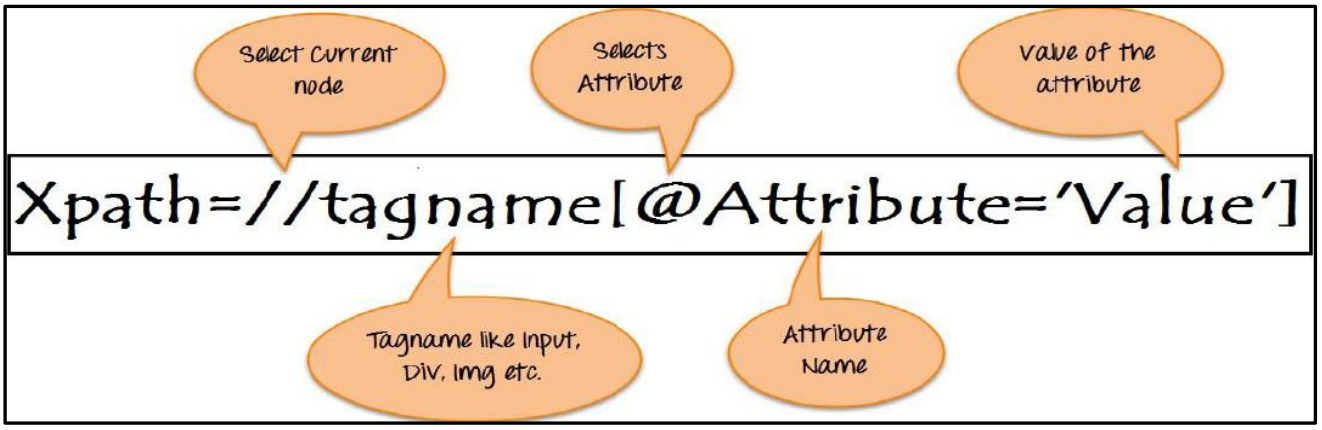
Use **Ctrl+F** to write XPath in the elements tab. XPath is used to locate the firstName tab. Based on the XPath syntax, first use // which means anywhere in the document. Here, **input** is the **tag name** that has an **id attribute** with the **value** "usernamereg-firstname".

XPath Syntax: XPath contains the path of the element situated at the web page. Standard Xpath syntax for creating XPath is.

Xpath=//tagname[@attribute='value'] Here,

- The XPath syntax generally starts with "//" double slash. That is to say, it will begin with the current node defined by the tag name.
- The next part is **tag_name**; it denotes the HTML tag name of the node.

- Subsequently, anything present inside the node encloses in the square brackets **[]**.
- Additionally, the "**@**" sign selects the attribute.
- Moreover, the "**Attribute_name**" is the name of the attribute of the node.
- And, the "**Value of attribute**" denotes the attribute value from the node.



Here, **Tagname can be tag names like input, Div, img, etc, Attribute can be any attribute like id, name, etc and Value is the value for the attribute.** The greatest advantage of using XPath for element identification is that it provides many different ways to locate same element. So, you can use different attributes and different strategies to identify an element using XPath. To identify an element using XPath, the syntax in Selenium

Syntax: driver.findElement(By.xpath("Element XPath expression"))

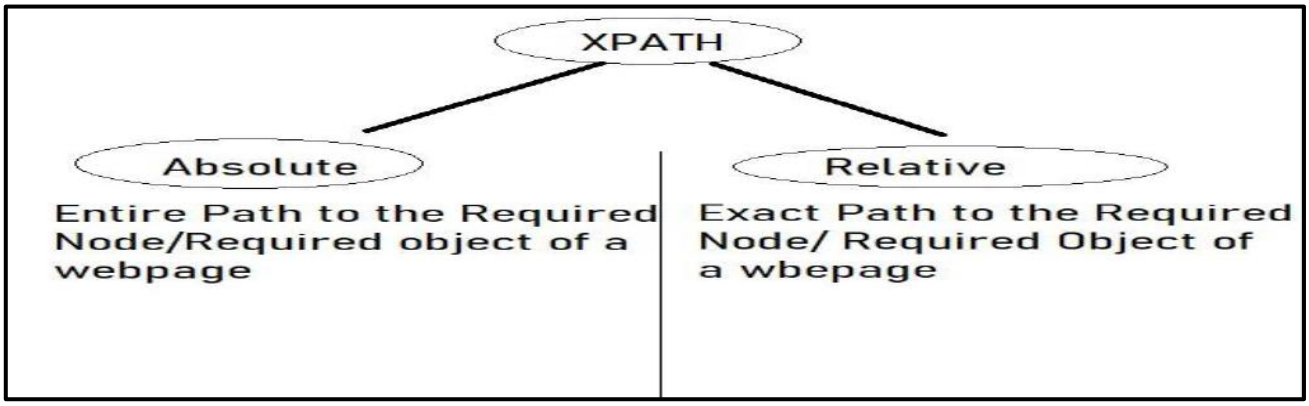
EX: driver.findElement(By.xpath(**//input[@name="firstname"]**))

To find the element on web pages accurately there are different types of locators:

XPath Locators	Find different elements on web page
ID	To find the element by ID of the element
Classname	To find the element by Classname of the element
Name	To find the element by name of the element
Link text	To find the element by text of the link
XPath	XPath required for finding the dynamic element and traverse between various elements of the web page
CSS path	CSS path also locates elements having no name, class or ID.

Types of X-Paths: There are two types of Xpath

1) **Absolute Xpath or Native XPath**
2) **Relative Xpath**

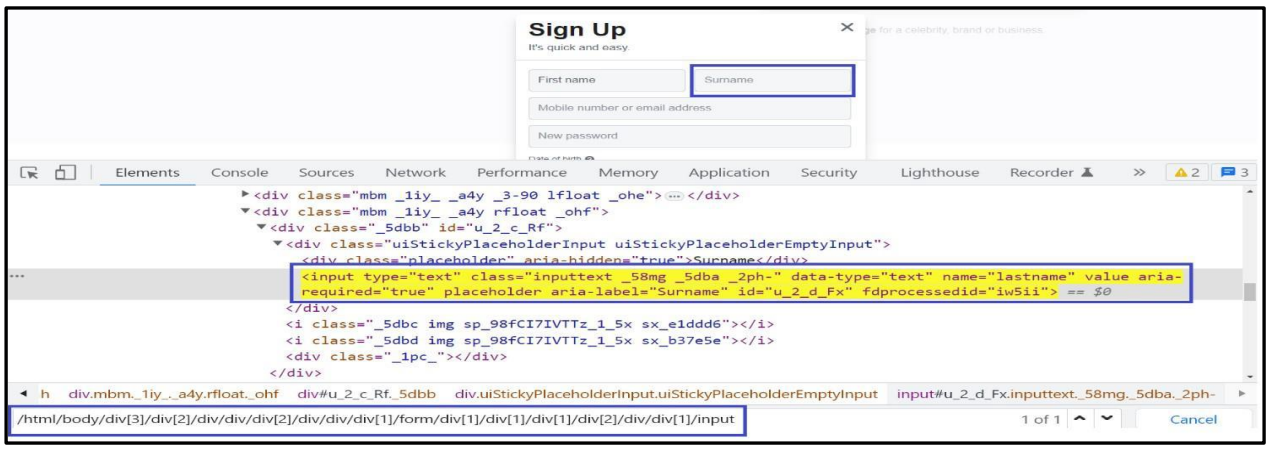


1) Absolute Xpath or Native Xpath: Absolute Xpath is the simplest form of XPath in Selenium. It is the **direct way to find the element**, but the disadvantage of the absolute XPath is that if there are any changes made in the path of the element then that XPath gets failed. The key characteristic of XPath is that it begins with the single forward slash (/) which means you can select the element from the root node. Absolute XPath uses the complete path starting from the root element to the desired element. The key characteristic of XPath is that it begins with the single forward slash(/) ,which means you can select the element from the root node.

Below is the **example** [https://www.facebook.com/] of an absolute Xpath expression of the element shown in the below screen.

/html/body/div[3]/div[2]/div/div/div[2]/div/div/div[1]/form/div[1]/div[1]/div[1]/div[2]/div/div[1]/input

/html/body/div[3]/div[2]/div/div/div[2]/div/div/div[1]/form/div[1]/div[1]/div[1]/div[2]/div/div[1]/input



Example:2 [https://demoqa.com /]

In the above document, if we want to find the absolute path of <img> node, then we will have to transverse starting from the root node(<html> node, as highlighted by the arrow in the above image). So, the resultant absolute XPath of the element will look like this:

/html/body/div/header/a/img

2) Relative Xpath: Relative Xpath starts from the middle of HTML DOM structure. It starts with double forward slash **(//)**. It can search elements anywhere on the webpage, means no need to write a long xpath and you can start from the middle of HTML DOM structure. Relative Xpath is always preferred as it is not a complete path from the root element. It is always preferred over an absolute XPath as it is not a complete path from the root element. We will locate the same element as in the case of absolute XPath.

For Relative XPath, the path starts from the middle of the HTML DOM structure. It begins with the double forward slash (//), which means it can search the element anywhere at the webpage.

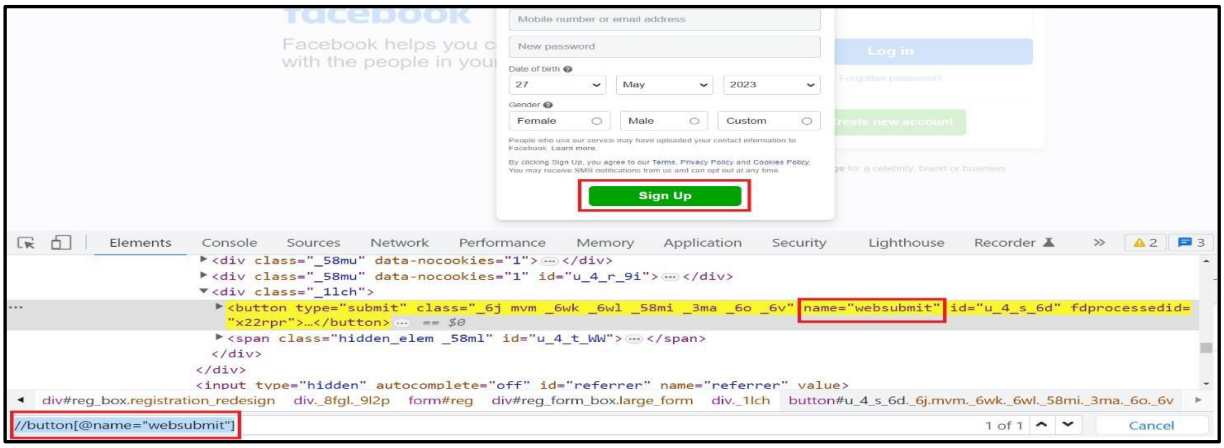
For example: //input[@id='ap_email'] Disadvantages:

Identifying the element will take longer because we specify a partial path rather than an exact path.

If multiple elements are on the same path, it will choose the first element identified.

Example: Now, let's understand this with the help of an example. I will launch Google Chrome and navigate to google.com and search https://www.facebook.com/. Here, I will try to locate the Sign-Up using XPath. On inspecting the web element, you can see it has a button tag and attributes like class and id. Now, I will use the tag name and these attributes

to construct XPath which in turn will locate the search bar. **//button[@name="websubmit"]**



Dynamic Xpath: Sometimes while working in a dynamic web environment, it becomes challenging to locate a particular web element by using general attributes such as name, class, etc. Sometimes, we may not identify the element using the locators such as id, class, name, etc. Several different elements may have similar properties, for example, similar names or class names. In those cases, we use XPath to find an element on the web page. At times, XPath may change dynamically and we need to handle the elements while writing scripts. The standard way of writing XPath may not work and we need to write dynamic XPath in selenium scripts. It is critical for automated testing to be able to recognize the web elements of an Application Under Test (AUT). Learning how to find web elements and writing dynamic XPath in Selenium manually may take a long time and a lot of practice.

Different ways of writing Dynamic XPath in Selenium with examples:

XPath Functions:

a) Using contains (): XPath Contains () is one of the methods used while creating an XPath expression. We can use it if part of the value of any attribute changes dynamically. It can identify any attribute by using its partial value. Contains () is a method used in XPath expression. It is used when the value of any attribute changes dynamically, for example, login information. The contain feature has an ability to find the element with partial text as shown in below XPath example.

In this example, we tried to identify the element by just using partial text value of the attribute. In the below XPath expression partial value 'sub' is used in place of submit button. It can be observed that the element is found successfully.

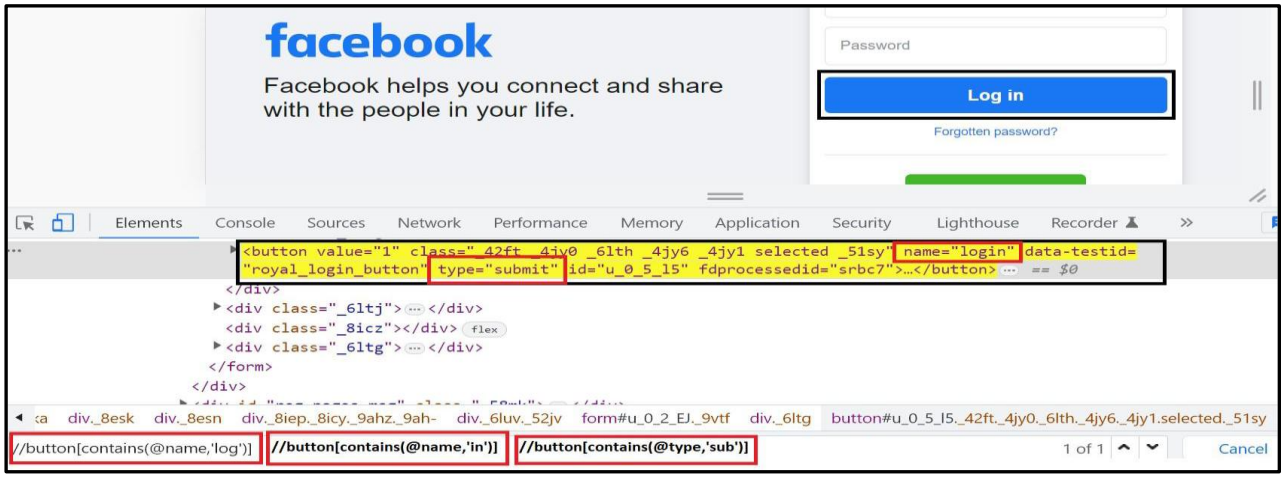
Complete value of 'Type' is '**submit'** but using only partial value '**sub'**.

Syntax: Xpath = **//*[contains(@type,'sub')]**

Complete value of 'name' is 'login' but using only partial value 'log'.

Example: Xpath= **//*[contains(@name,'log')]**

In the above expression, we have taken the 'name' as an attribute and 'log' as a partial value as shown in the below screenshot.



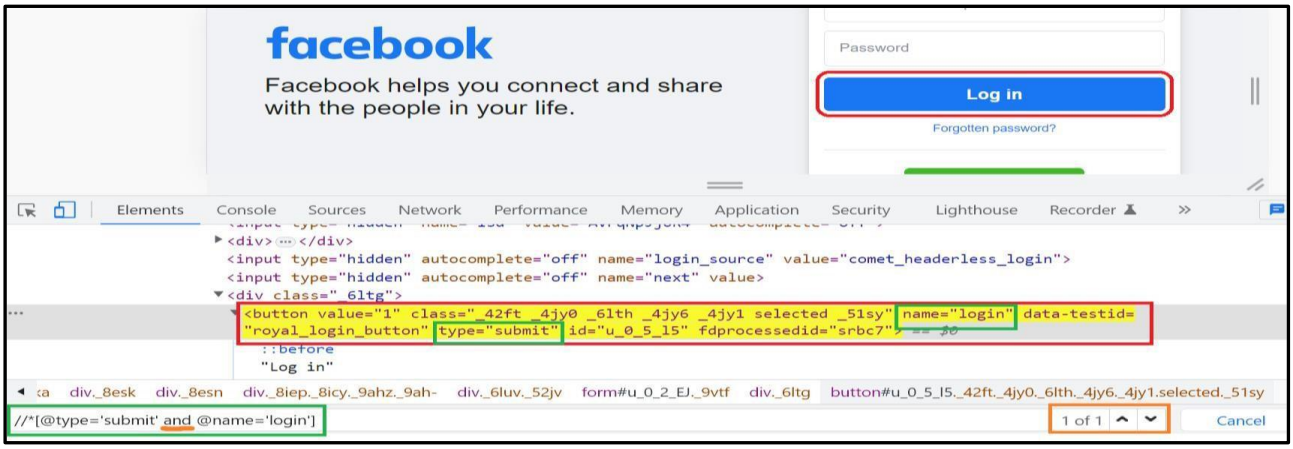
b) Using AND & OR: The "**AND**" operator is used to combining two different conditions or attributes to identify any element from a webpage using XPath efficiently. In **AND** expression, two conditions are used, both conditions should be **true** to find the element. It fails to find element if any one condition is **false**.

Syntax: //tag_name[@name = 'Name value' and @id = 'ID value']

Here, we are using name and id attribute; you can use any attribute required to identify the element uniquely.

Example: Xpath=//input[@type='submit' and @name='btnLogin']

In below expression, highlighting 'LOGIN' element as it having both attribute 'type' and 'name'.



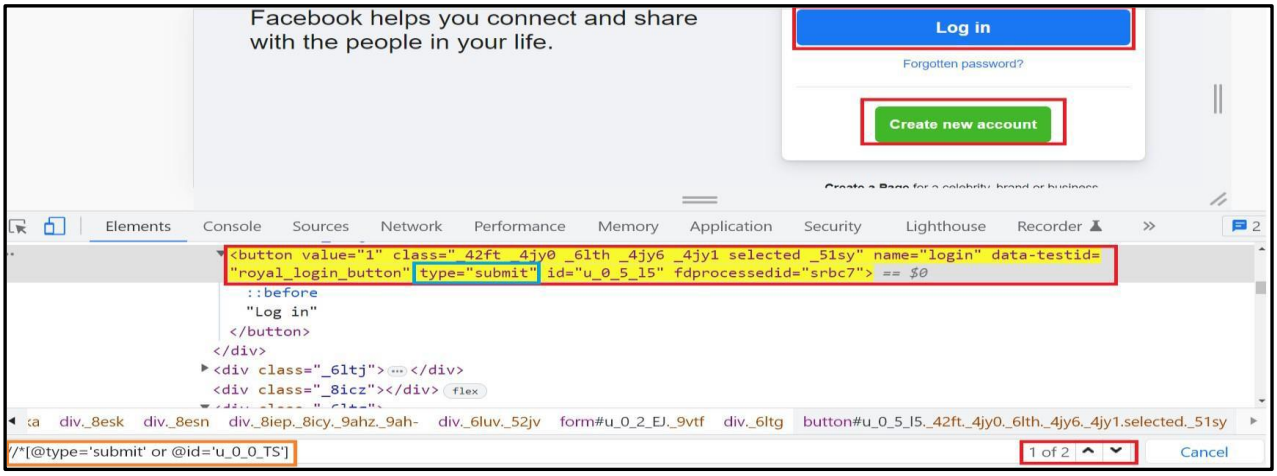
In **OR** expression, two conditions are used, whether 1st condition OR 2nd condition should be true. It is also applicable if any one condition is true or maybe both. Means any one condition should be true to find the element.

In the below XPath expression, it identifies the elements whose single or both conditions are true.

Syntax: //tag_name[@name = 'Name value' or @id = 'ID value']

Example: Xpath= //*[@type='submit' or @id='u_0_0_TS']

Highlighting both elements as "Login "element having attribute 'type' and "Create new account" element having attribute 'id'.

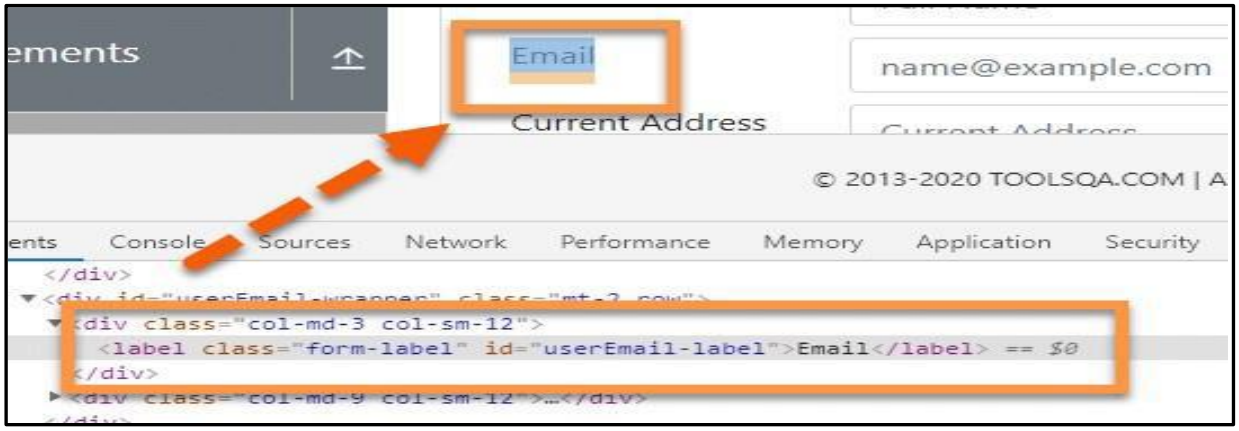


c) Using Text (): This function uses the text of the web element for locating the element on the webpage. This function is quite useful if your element contains the text, for example, labels, which always contain a static text. The XPath text() function is a built-in function of selenium webdriver which is used to locate elements based on text of a web element. It helps to find the exact text elements and it locates the elements within the set of text nodes. The elements to be located should be in string form.

Syntax: //tag_name[text()='Text of the element']

Let's have a look at the webpage "**https://demoqa.com/text-box**" and try to locate the "Email " label using text() function. We will be using the text present in the UI for recognizing the element. The text on the UI is "Email", and the same is present between element tag, i.e., <label> ….</label>

Example1: //label[text()="Email"]



In our case, we find the element with text "**'Facebook helps you connect and share with the people in your life.**". A built-in method in Selenium WebDriver that is used with XPath locator to locate an element based on its exact text value. Using XPath- text() method, we can write the Java code along with the dynamic XPath location as: findElement(By. xpath("//*[text()='Facebook helps you connect and share with the people in your life.']");"text() method" is used to identify an element based on the text available on the web page.

Example2: Xpath=//*[text()='Facebook helps you connect and share with the people in your life.']

d) Using starts-with(): The starts-with() function in XPath is used to select elements whose attribute value starts with a specified string. It can be very useful when working with web elements in Selenium with Java. starts-with() is a method that finds those elements whose attribute value changes on refresh on the web page. This method checks thestarting text of an attribute and finds lements whose attribute changes dynamically."starts-with()" is used to identify an element, when we are familiar with the attributes value (starting with the specified text) of an element.

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

- tagname refers to the HTML tag name of the element you want to select.
- @attribute represents the attribute of the element that you want to check.
- 'value' is the string you want to check if the attribute value starts with.

Example1: //*[starts-with(@id,'ema')]

XPath Axes Methods: As we know that path defines the location of a node using absolute or relative path. In the same manner XPath axes are used to identify elements by their relationship like parent, child, sibling, etc. Axes refer to axis on which elements are lying relative to an element. These XPath axes methods are used to find the complex or dynamic elements. XPath axes in Selenium are methods to identify those dynamic elements which are not possible to find by normal XPath method such as ID, Classname, Name, etc. Axes are so named because they talk about the axis on which elements are lying relative to an element. Dynamic web elements are those elements on the webpage whose attributes

dynamically change on refresh or any other operations. The commonly useful XPath axes methods used in Selenium WebDriver are **child, parent, ancestor, sibling, preceding, self, namespace, attribute, etc**. XPath axes help to find elements based on the element's relationship with another element in an XML document. XML documents contain one or more element nodes. They are considered as trees of nodes. If an element contains content, whether it is other elements or text, then it must be declared a start tag and an end tag. The text defined between the start
tag and end tag is the element content. The topmost element of the tree is called root element. An example of a basic HTML page is shown below screenshot.

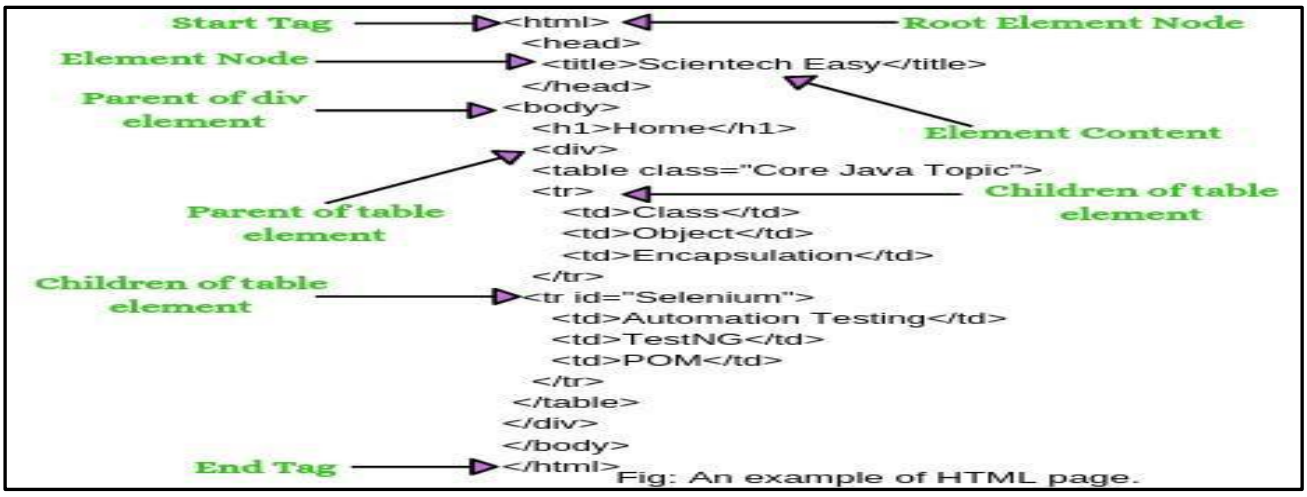


Fig: An example of HTML page.

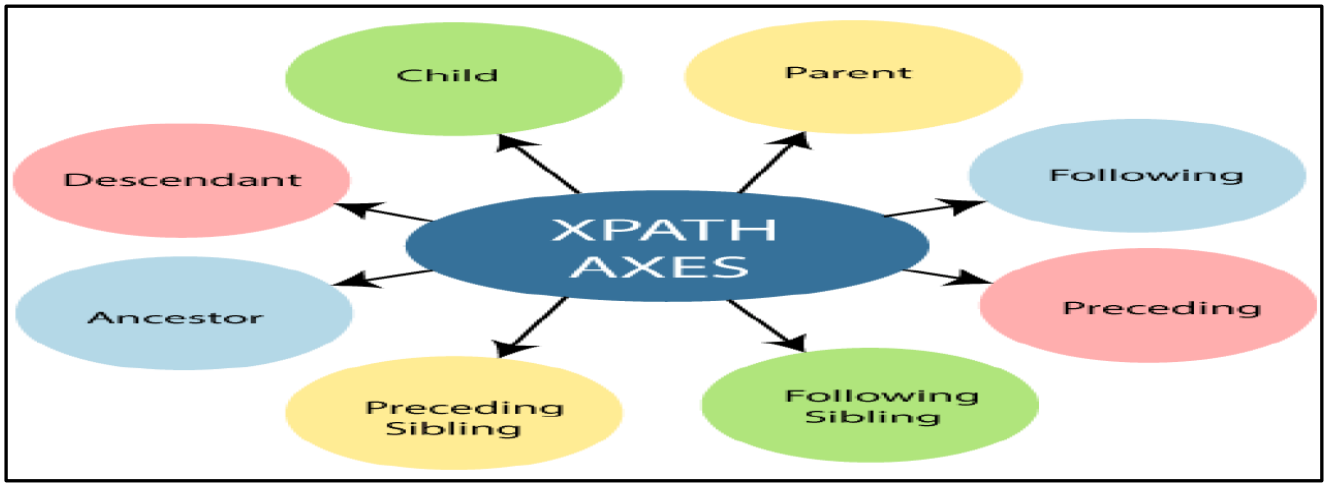
XPath axes methods in Selenium are used to navigate through the elements in the HTML document based on their relationships with other elements. There are several axes methods available in XPath:

XPath Terminology: Nodes, Atomic values, Parents, Children, Siblings.

i. **Nodes:** DOM represents trees of nodes. The topmost element of the tree is called root node. The example of nodes in the XML document above: "**html**" is the root of element node, "**title**" is an element node. XPath contains seven kinds of nodes: **element, attribute, text, namespace, processing-instruction, comment, and document nodes**.
ii. **Atomic values:** The node which has **no parents or children** is called atomic values. For example, Automation Testing, TestNG, POM.
iii. **Parents:** Each element and attribute has one parent like **father or mother**. For example, "**body**" is the parent of **div** element. "**div**" is the parent of the **table** element.
iv. **Children:** Element nodes that may contain **zero, one, or more children**. For example, **tr** is children of **table** element, **div** is the children of **body** element, **table** is children of **div** element.
v. **Siblings:** The node that has the **same parent** is called siblings. In the above XML document, **title** and **body** elements both are siblings.

Syntax for XPath axes: //tagname/axes::target elements

XPath Axes Methods in Selenium:



Name	Functionality
parent	It identifies the current node's parent. **//div[@class='Jet']//parent::div**
child	It selects the children of the current node that are on the same level. **//div[@class='ByAir']//child::div[@class='Jet']/h4**
ancestor	It is the current node's ancestor (grandparents, parents, etc.) till the root node is selected. **//div[@class='ByAir']//ancestor::div[@class='TransportingVehicles']**
descendant	It selects all the current node's descendants until the leaf node has no children. **//div[@class = 'signup_form new']//descendant-or-self::input[2].**
following	It selects all the incoming nodes after the current node, except the attribute and descendants. **//input[@id = 'u_0_r']//following::input[2]**
following-sibling	It selects all the same level nodes after the current node, having the same parent. **//input[@id = 'u_0_5']//following-sibling::label.**
preceding	It selects all the previous nodes from the current node except the attribute and ancestors. **//div[@class='Jet']//preceding::div**
preceding-sibling	It selects all the same level nodes before the current node, having the same parent. **//a[text() = 'Videos']//preceding-sibling::a[1]**
self	It selects the current node **//input[@id = 'u_0_3']//self::input**

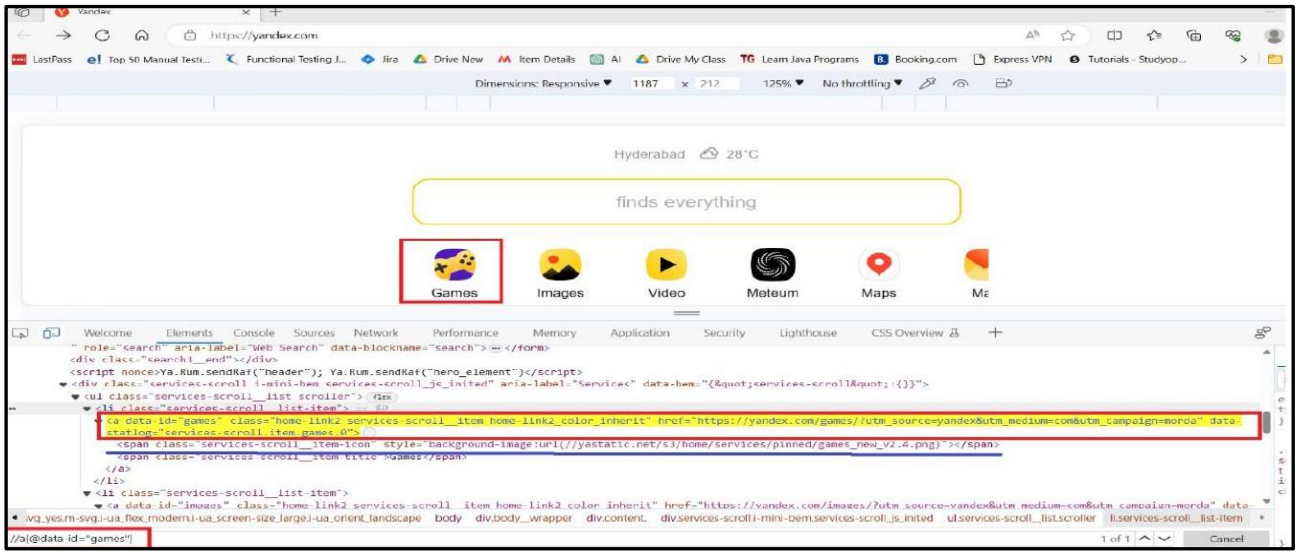
A) Child: The child axis is one of the XPath axes that contains all the child nodes of the current context. It selects all children elements of the current node. Child Axis in XPath is used to find all the child nodes of the current node. Child axes are used to traverse all child

elements of the current html tag. As the name specifies, this approach is used to locate all the child elements of a particular node.

Syntax: //tag[@attribute='value']//child::childtagname

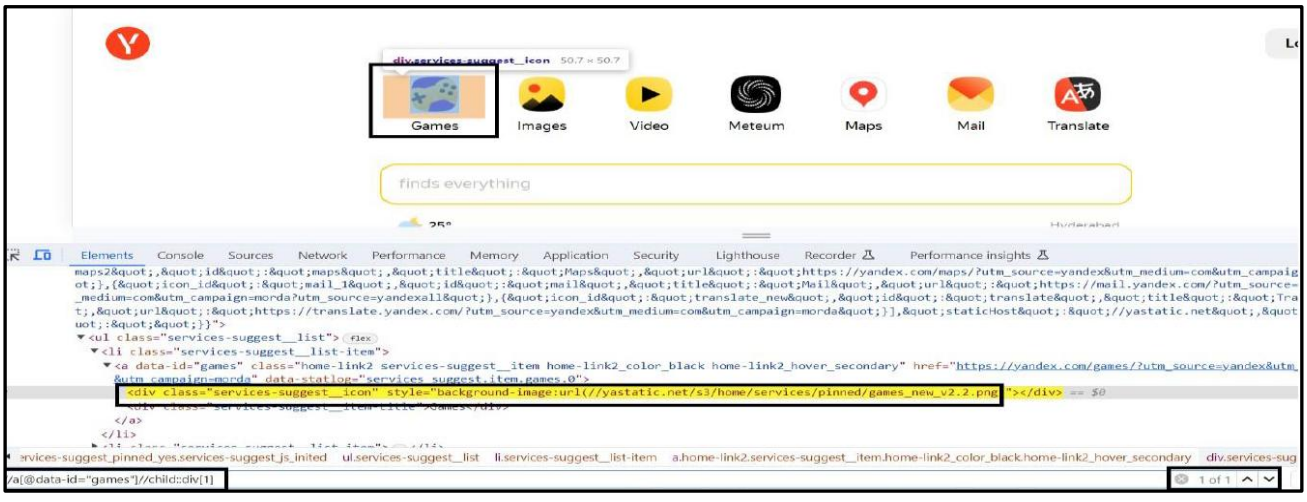
Example: //a[@data-id="games"]//child::div[1]

Now let us seee one real time example on finding the child web element. Open webpage [https://www.yandex.com], right-click on Yandex.in and go to inspect option as shown below screenshot.



First, we will find XPath of the current node. **//a[@data-id="games"]**

Now we will find out XPath of children elements of current node using child axis. This expression identified 2 children nodes using the child axis. We can get the XPath of different children elements according to the requirement by putting [1],[2].



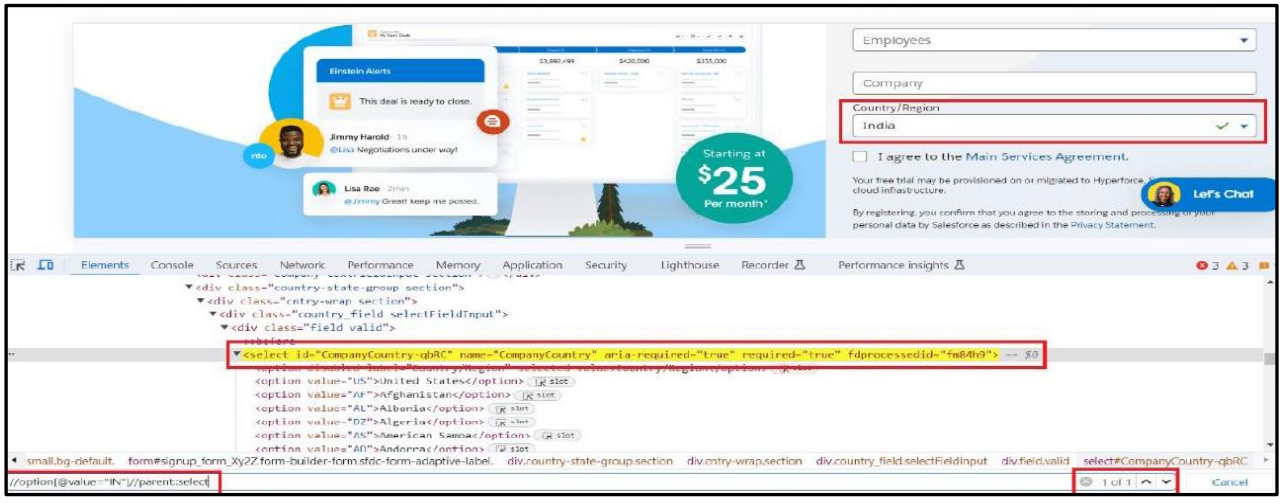
```
package XPathAxis;

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

public class ChildAxis {

    public static void main(String[] args)
    {
System.setProperty("webdriver.chrome.driver","C:\\Automation 12pm Class New\\chro

        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://www.guru99.com/");

        driver.findElement(By.xpath("//*[@class=\"srch\"]//child::span[5]")).clic

    }
```

B) Parent: The parent axis selects the parent of the current node. The parent node may be either root node or element node. The root node has no parent. Therefore, when the current node is root node, the parent axis is empty. For all other element nodes, the parent axis contains a maximum of one node. Parent axes are used to traverse parent element of the current html tag. Parent in Selenium is a method used to retrieve the parent node of the current node selected in the web page. It is very useful in the situation when you select an element and need to get the parent element using Xpath. This method is also used to get the parent's parent.

Syntax: //*[attribute='value']/parent::parenttagname

Example: //option[@value="IN"]//parent::select

Now let us seee one real time example on finding the parent web element. Open webpage [https://www.salesforce.com/form/signup/elf-v2-login/?d=70130000000Enus], right-click on it and go to inspect option as shown below screenshot.

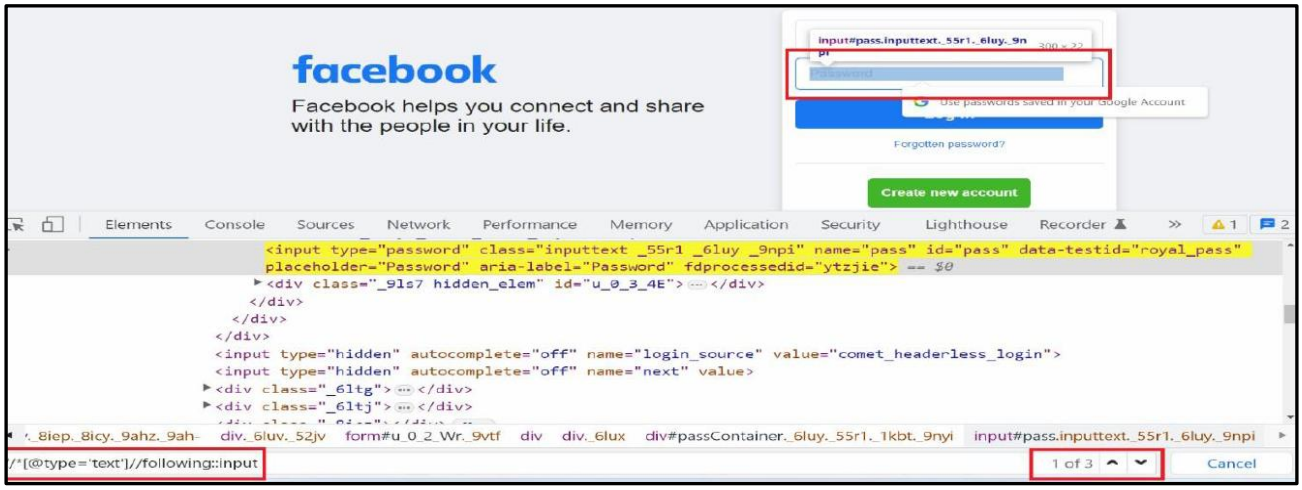


First, we will find XPath of the current node. **//option[@value="IN"]**

Now we will find out XPath of parent elements of current node using parent axis.

//option[@value="IN"]//parent::select

Once after finding the XPath of the signup button using parent axes, the code snippet of your program will be:




```
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://www.salesforce.com/form/signup/elf-v2-login/?d=70130000000Enus");
    //driver.findElement(By.xpath("//option[@value=\"AU\"]//parent::select")).click();
Select drpCountryRegion = new Select(driver.findElement(By.xpath("//option[@value=\"AU\"]//parent::select")));
//drpCountry.selectByVisibleText("ZIMBABWE");
//drpCountryRegion.selectByVisibleText("United States");
//drpCountryRegion.selectByVisibleText("Poland");
drpCountryRegion.selectByVisibleText("Zimbabwe");
```

C) Following: This method selects all the elements in the HTML document from the current node. The following axis in XPath is a powerful tool for navigating XML trees in Selenium tests. It selects all the nodes in the document, post the closing tag of the current node, no matter where they are nested or at what level. Let's say you have a web page with multiple sections and want to identify an element that appears after a particular section. With the following axis, you can effectively locate that element without navigating through the entire tree structure.

Here's an example of how you might use the following axis in a Selenium script:

Xpath=//*[@type='password']//following::input

Xpath=//*[@type='password']//following::input[2]

In the above code would select all input tags in the document.

D) Ancestor: The ancestor axis method in XPath is particularly useful when dealing with complex XML documents or web pages with deeply nested elements. This method allows you to select all ancestor elements (parent, grandparent, others) of the current node in reverse document order (from the closest ancestor to the furthest).

Below is an example.

Syntax: //tagname[@attribute='value']//ancestor::ancestorTag

Xpath=//*//ancestor::p

Xpath=//*//ancestor::div[2]

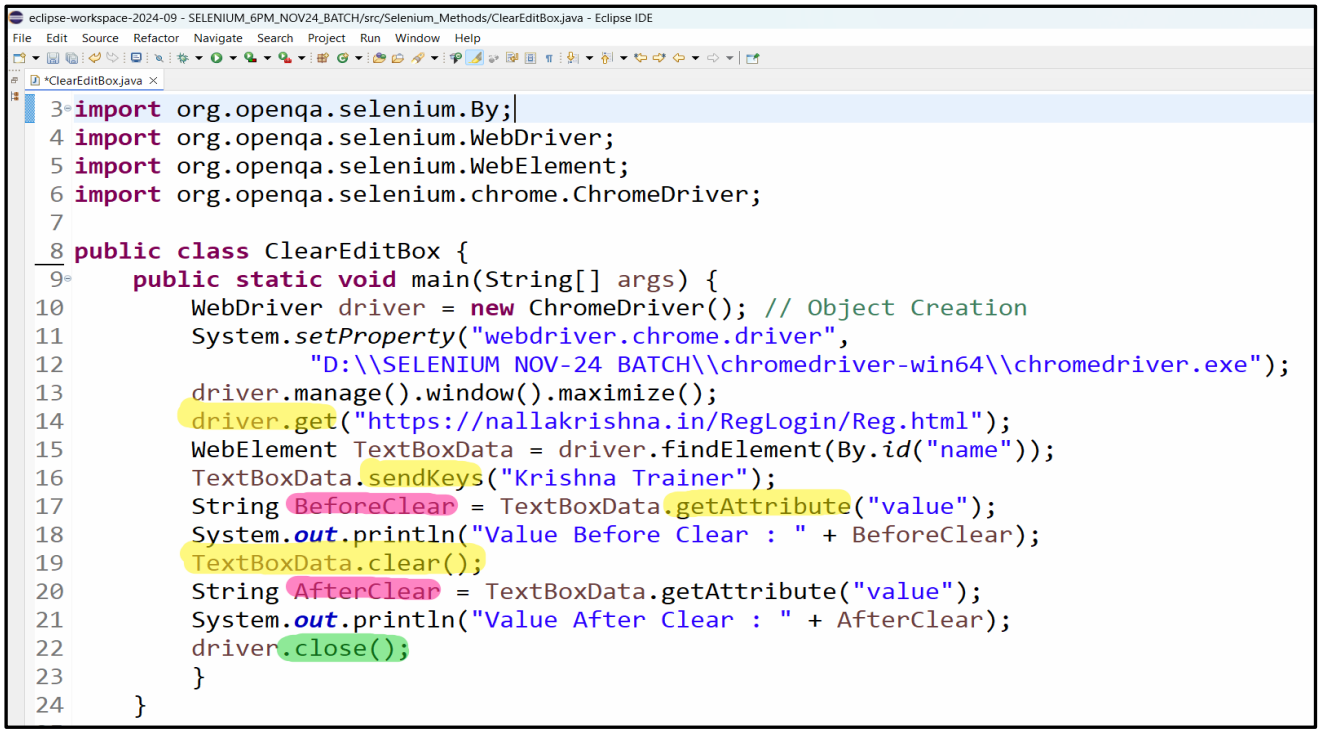
****

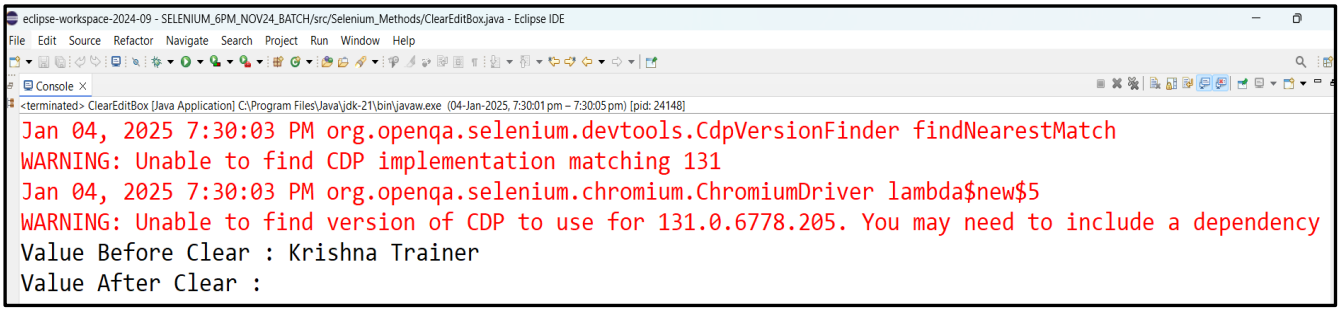
Selenium clear() Method

Selenium clear() method is used to clear text from input textbox or text area. Clear() method required when you enter some text in textbox or some text is already present in textbox and you want to erase those texts to type new text in it. This method clears the edit box and also makes the field enabled.

Textboxes are the input boxes in HTML that are used to take input from the users. Text boxes are widely used in web forms for taking inputs such as email ID, password, feedback, mobile numbers, and other information in text. While we are dealing with web forms it's very common to interact with text boxes and sometimes the textbox, we want to interact with is already filled with some default values, so we have to clear out the content of the text box to use it. clear method is used to clear text of any field. Clear() method is used to clear or erase text from textbox or text area web elements.

For verifying how clear() method ,First of all we need to identify the element with help of any of the locators like id, class, name, xpath or css and then apply sendKeys() method to type some text inside it. Next we shall apply the clear() method on it. To check if the edit box got cleared, we shall use getAttribute() method and pass value parameter as an argument to the method. We shall get blank value after clear() is applied.





****

JavaScriptExecutor

JavaScriptExecutor is an interface provided by Selenium that helps in executing JavaScript commands. This interface provides methods to run JavaScript on the selected window or the current web page. It is available for all the language bindings supported by Selenium. In Selenium Webdriver, locators like XPath, CSS, etc. are used to identify and perform operations on a web page. In case, these locators do not work you can use JavaScriptExecutor. You can use JavaScriptExecutor to perform a desired operation on a web element.

The JavaScriptExecutor in Selenium can be used directly by importing the following package in the automation test scripts:

import org.openqa.selenium.JavascriptExecutor;

JavascriptExecutor Provides Two Methods:

A. ExecuteScript **B**. ExecuteAsyncScript

A) executeScript():

This method executes JavaScript in the context of the currently selected window or frame in Selenium. The script will be executed as the body of an anonymous function and the script can return values.

Data types returned are: Boolean Long String List WebElement.

```
JavascriptExecutor js = (JavascriptExecutor) driver;
js.executeScript("window.scrollBy(0,100)", "");
```

B) ExecuteAsyncScript

This method is used to execute the asynchronous Javascript in the current window or frame. An asynchronous Javascript executed is a single thread while the rest of the page continuous parsing. which enhances the performance.

How to get started with JavascriptExecutor

→ Import the package
→ Create a reference
→ Call the JavascriptExecutor methods

```
//importing the package
Import org.openqa.selenium.JavascriptExecutor;
//Creating a reference
JavascriptExecutor js = (JavascriptExecutor) driver;
//Calling the method
js.executeScript(script, args);
```

****

Scroll Bar [Scroll a Page]

A Scrollbar is a lets you move around screen in horizontal or vertical direction if the current page scroll does not fit the visible area of the screen. It is used to move the window up and down. Selenium Webdriver does not require scroll to perform actions as it manipulates DOM. But in certain web pages, elements only become visible once the user have scrolled to them. In such cases scrolling may be necessary.

Selenium WebDriver offers multiple methods for seamless scrolling, providing diverse techniques to simulate this action.

1) scrollBy() method: This scrollBy() method is handy when adjusting the view by a specific number of pixels in Selenium WebDriver. Similar to the scrollTo() method, it also takes (x, y) coordinates as arguments. However, in the scrollBy() method, the coordinates signify the number of pixels to scroll rather than the absolute position to scroll to. For instance, to smoothly scroll 50 pixels on the x-axis and 100 pixels on the y-axis, you would provide these values as arguments to the method.

Ex: js.executeScript("window.scrollBy(50,100)");

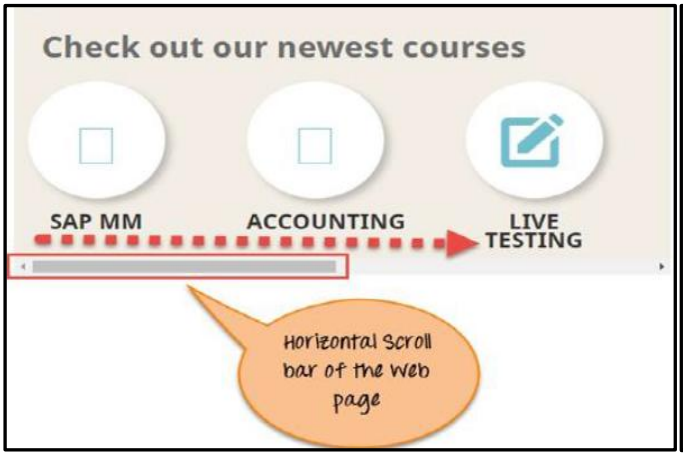
2) scrollIntoView() method: This scrollIntoView() method in Selenium WebDriver is a valuable approach for scrolling to a specific WebElement that is currently not visible on the screen. This method takes the script to scroll and the WebElement as arguments. Like the scrollTo() and scrollBy() methods, it effectively manages elements' visibility during automation testing.

Ex: js.executeScript("arguments[0].scrollIntoView();", element);

3) scrollTo() method: This scrollTo() method is valuable for scrolling a web page to a specific location using Selenium WebDriver. This method requires the (x, y) coordinates of the desired location as arguments. For instance, to effortlessly scroll down to the bottom of the web page, you would pass the y-coordinate set as the maximum height of the document (web page).

Ex: js.executeScript("window.scrollTo(0, document.body.scrollHeight)");

Scroll Bar is of two types: Horizontal and Vertical Scroll bar as shown in below screenshot.



Scrolling is one of the most important features of any application. You will not find any website which does not have scrolling capability. It's almost impossible to display all the

content in a single frame, users need to scroll down or up to get all the content of the application. Selenium WebDriver directly interacts with the DOM[Document Object Model], and several operations can be carried out without the need to scroll. However, there are rare instances where specific web elements are only accessible after the user scrolls to the element. In these situations, automating the scrolling process is required.

To scroll down the web pages we use the JavaScriptExecutor. The javascriptexecutor is an interface that enables to running of JavaScript methods from Selenium scripts. A Scroll is a JavaScript method. The JavaScriptExecutor provides an interface that enables QAs to run JavaScript methods from Selenium scripts. Hence, to scroll up or down with Selenium, a JavaScriptExecutor is a must. JavaScriptexecutor is an interface that is used to execute JavaScript through selenium webdriver. The JavaScript Executor is a powerful feature of Selenium WebDriver that allows you to execute JavaScript code directly in the browser.

Syntax: JavaScriptexecutor js = (JavascriptExecutor) driver;
js.executeScript("window.scrollBy(0,350)", "");

ScrollBy is a JavaScriptexecutor method that accepts two parameters x and y, which defines horizontal and vertical axes.

Scroll down a web page using the selenium webdriver with following Scenarios:

A) Scroll a web page Horizontally and Vertically: In Selenium, you can use the JavaScript Executor to scroll a web page both horizontally and vertically. In order to scroll the page both horizontally and vertically, you need to use the "scrollBy" method twice in a row, once for horizontal scrolling and once for vertical scrolling. To scroll down a page we have to use the JavaScript command window.scrollBy. Then pass the pixel values to be traversed along the x and y axis for horizontal and vertical scroll respectively. Here is an example of how to use the "scrollBy" method to scroll a web page both horizontally and vertically in Selenium.

```
JavaScriptexecutor js = (JavascriptExecutor) driver;
```

```
js.executeScript("window.scrollBy(0,300)");  // Vertical scroll(Top to Bottom)
js.executeScript("window.scrollBy(0,-300)");  // Vertical Scroll (Bottom to Top)
```

```
js.executeScript("window.scrollBy(300,0)"); // Horizontal Scroll(Left to Right)
js.executeScript("window.scrollBy(-300,0)"); // Horizontal Scroll(Right to Left)
```

B) Scrolling to a specific WebElement: Scroll based on the visibility of the Web element. Before you perform scroll operations in selenium, it is necessary that the corresponding web element is present under DOM. This scrollIntoView() method in Selenium WebDriver is a valuable approach for scrolling to a specific WebElement that is currently not visible on the screen. This method takes the script to scroll and the WebElement as arguments. When called on an element, it will scroll the element's parent container so that the element becomes visible in the viewport.ScrollIntoView() method will help you to scroll to the desired element.

```
JavascriptExecutor js = (JavascriptExecutor) driver;
js.executeScript("arguments[0].scrollIntoView();", webElement);
```

This functionality becomes particularly relevant when seamlessly integrating the scroll down in the Selenium WebDriver technique.

C) Scroll down to the Bottom and Top of the page: In Selenium, you can use the JavaScript Executor to scroll down to the bottom of a web page and Scroll up to the top of the page. You can use the "scrollTo" method to scroll to the bottom and top of the page. The following code snippet shows an example of how to scroll down to the bottom of a web page and top of a web page using Selenium in Java.

```
<JavaScriptexecutor js = (JavascriptExecutor) driver;
js.executeScript("window.scrollTo(0, document.body.scrollHeight)");
```

In the above example, the JavaScript Executor is first cast to a JavascriptExecutor object and then the "executeScript" method is used to execute a JavaScript code. The JavaScript code passed to the "executeScript" method is "window.scrollTo(0, document.body.scrollHeight)", which scrolls the page to the bottom.

```
<JavaScriptexecutor js = (JavascriptExecutor) driver;
js.executeScript("window.scrollTo(document.body.scrollHeight,0)");
```

In the above example, the JavaScript Executor is first cast to a JavascriptExecutor object and then the "executeScript" method is used to execute a JavaScript code. The JavaScript code passed to the "executeScript" method is "window.scrollTo(document.body.scrollHeight,0)", which scrolls the page to the top.

D) Scroll down the web page by Pixel: With the help of JavaScriptExecutor, you can scroll to a particular point by specifying the exact pixel or coordinates of the object. This can be achieved by using the 'ScrollBy' function.in JavaScript.

```
JavascriptExecutor js = (JavascriptExecutor) driver;
//specify the number of pixels the page must be scrolled
js.executeScript("window.scrollBy(0,3000)");
```

E) Scroll **web page using Actions Class:** To achieve this, it is necessary to locate the WebElement representing the scroll bar and execute the scrolling action. In Selenium, you can use the sendKeys() method to simulate key presses, including the "Page Down" and "Page Up".

```
WebElement element = driver.findElement(Locator)));
Actions act = new Actions(driver);
act.sendKeys(element ,Keys.PAGE_DOWN).build().perform();    //Page Down
act.sendKeys(element , Keys.PAGE_UP).build().perform();       //Page Up
act.sendKeys(element , Keys.END).build().perform();
```

****

Screenshot's in Selenium

The whole point of automated testing is defeated if one has to re-run an entire test every time a script fails. If something goes wrong, it helps to have the bug pointed out in the code and to have some visual representation of the exact anomaly. Similarly, a tester must check if the application's flow is as intended. Once again, visual representation through screenshot testing helps in this case.

Selenium webdriver can automatically take screenshots during the execution. But if users need to capture a screenshot on their own, they need to use the TakeScreenshot method which notifies the WebDrive to take the screenshot and store it in Selenium. If a test fails, the ability to take screenshot in Selenium is especially helpful in understanding what went wrong. A Screenshot in Selenium Webdriver is used for bug analysis. To capture screenshots in Selenium, you should use the TakesScreenshot method. This method notifies WebDriver that it should take a screenshot in Selenium and store it.

WebDriver allows you to execute your tests against different browsers and enables you to use a programming language in creating your test scripts. To save the Selenium screenshot in the desired location, you can use the getScreenshotAs. To take a screenshot in Selenium, we use an interface called TakesScreenshot, which enables the Selenium WebDriver to capture a screenshot and store it in different ways. It has a got a method "getScreenshotAs() " which captures the screenshot and store it in the specified location.

Syntax: File file = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
Example: File s = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

In the above code, we convert the WebDriver object (driver) to TakeScreenshot. And call getScreenshotAs() method to create an image file by providing the parameter *OutputType.FILE.

Syntax: FileUtils.copyFile(file, new File("path\\imagename.png"));
Example: FileUtils.copyFile(s, new File ("C:\\Automation3pmClass\\Screenshots\\Screen01.png"));

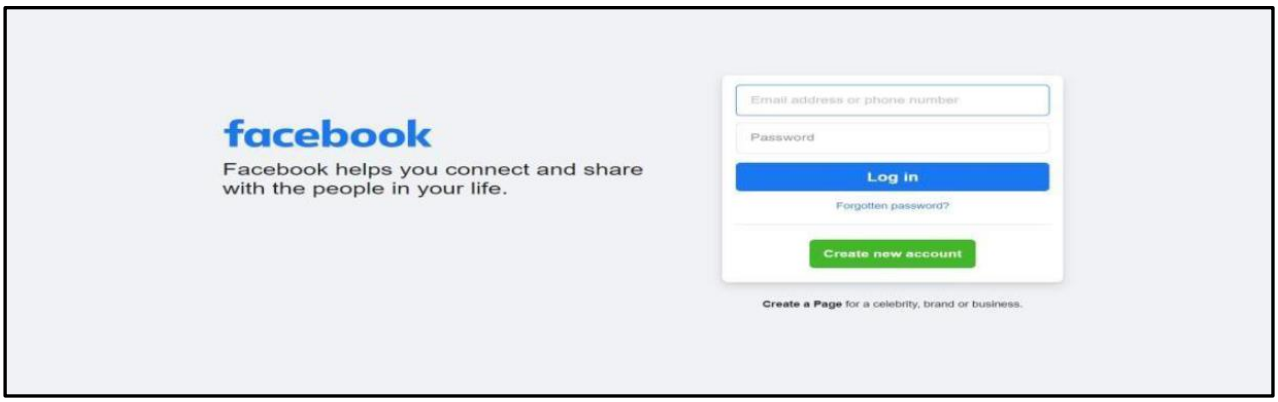
Example Program:

```
package ScreenShot;
import java.io.File;
import java.io.IOException;
import org.apache.commons.io.FileUtils;
import org.openqa.selenium.By;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
public class FB_ScreenShot    {
```

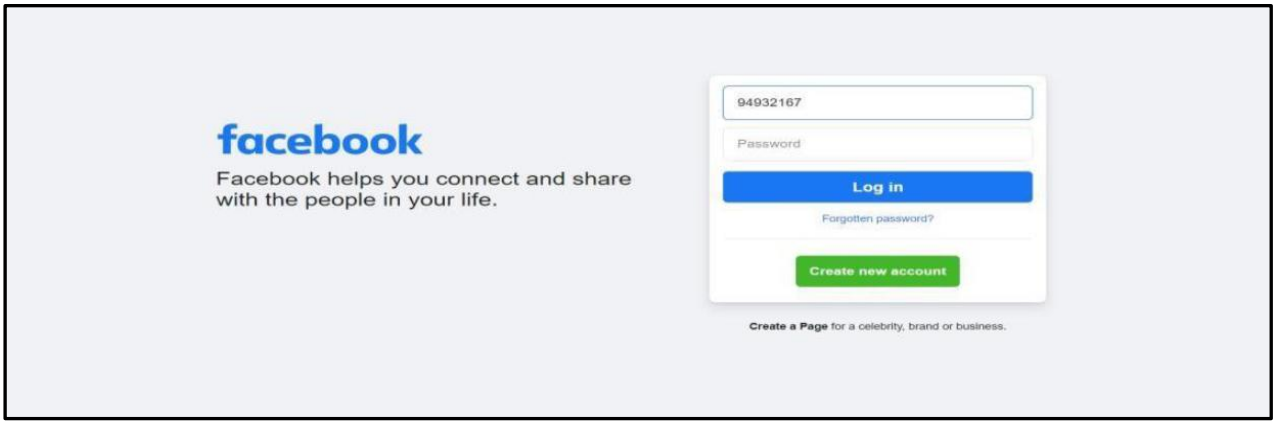
```
public static void main(String[] args) throws InterruptedException, IOException  {
System.setProperty("webdriver.chrome.driver","C:\\Selenium2023\\chromedriver_win3
2(1)\\chromedri ver.exe");
WebDriver driver = new ChromeDriver(); Thread.sleep(4000);
driver.manage().window().maximize(); Thread.sleep(4000);
driver.get("http://www.facebook.com");
String baseUrl = "http://www.facebook.com"; //String sname="Krishna";
Thread.sleep(4000); String tagName = ""; driver.get(baseUrl);
tagName = driver.findElement(By.id("email")).getTagName();
System.out.println(tagName); Thread.sleep(4000);
tagName = driver.findElement(By.id("pass")).getTagName();
System.out.println(tagName); Thread.sleep(4000);
tagName = driver.findElement(By.name("login")).getTagName();
System.out.println(tagName);
Thread.sleep(4000);
//Screen Shot-1
File s = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(s, new File("C:\\Automation3pm
Class\\Screenshots\\Screen01.png"));
driver.findElement(By.id("email")).sendKeys("94932167");
// phoneNumber.sendKeys("9493217"); Thread.sleep(4000);
//Screen Shot-2
File s1 = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(s1, new File("C:\\Automation3pm
Class\\Screenshots\\Screen02.png"));
driver.findElement(By.name("pass")).sendKeys("Krishna****");
//phoneNumber.sendKeys("Krishna****"); Thread.sleep(4000);
//Screen Shot-3
File s2 = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
```

```
FileUtils.copyFile(s2, new File("C:\\Automation3pm
Class\\Screenshots\\Screen03.png")); WebElement login =
driver.findElement(By.name("login")); login.click();
Thread.sleep(4000);
//Screen Shot-4
File s3 = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(s3, new File("C:\\Automation3pm
Class\\Screenshots\\Screen04.png")); driver.close();
}       }
```

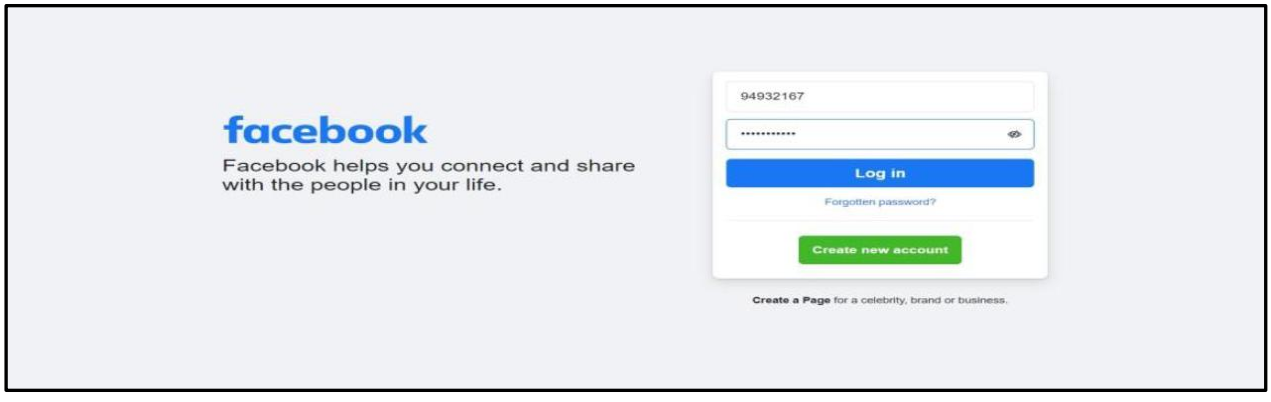
ScreenShots: ***ScreenShot-1***



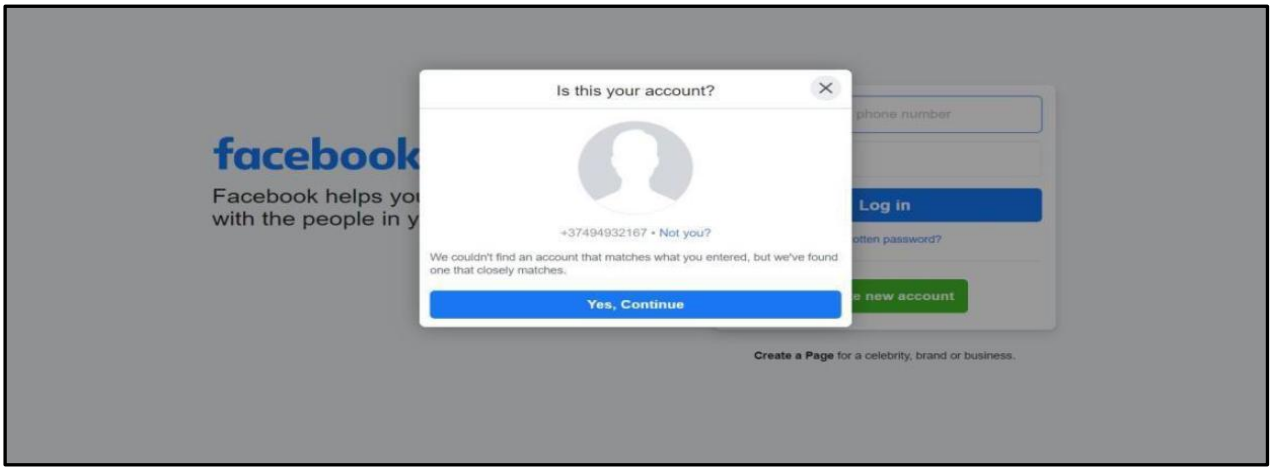
ScreenShot-2



ScreenShot-3



ScreenShot-4

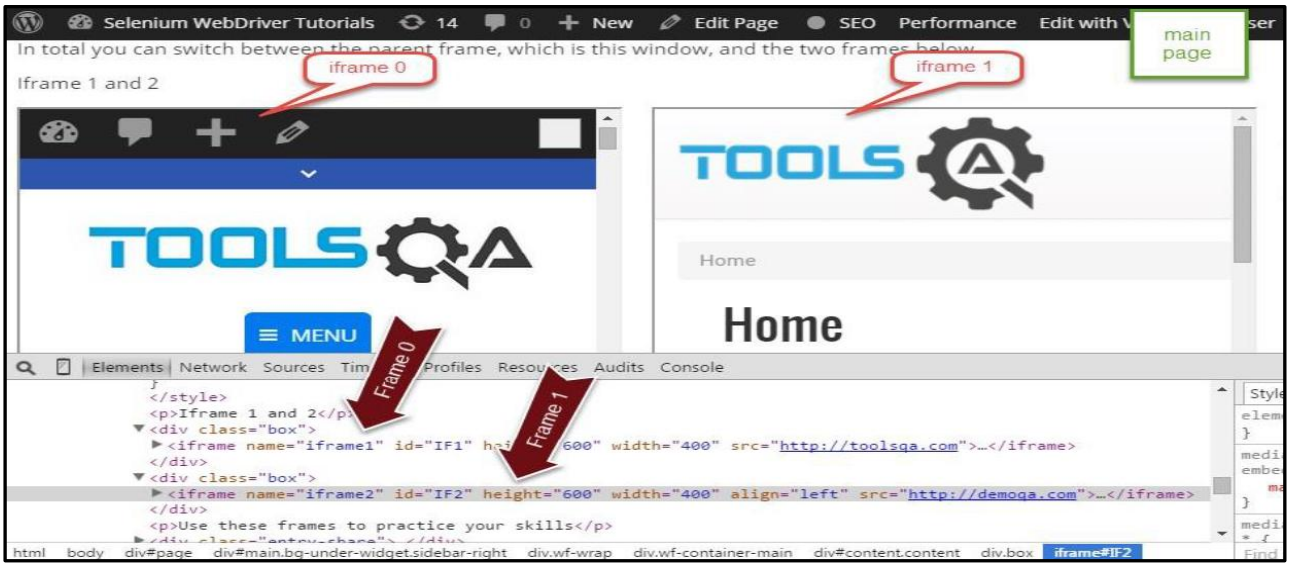


****

Frames in Selenium

The frame enables a developer to split the screen horizontally or vertically by using the frameset tag. The iframes are mainly used to insert content from external sources. For example, an advertisement displayed on a web page. They can float within the webpage, which means one can position an iframe at a specific position on a web page. An iframe is also known as an inline frame. The frame enables a developer to split the screen horizontally or vertically by using the frameset tag. iFrame is a HTML document embedded inside an HTML document. iFrame is defined by an <iframe></iframe> tag in HTML. With this tag, you can identify an iFrame while inspecting the HTML tree.

You can find iFrame's test page here https://demoqa.com/frames We will use this to learn iFrame handling logic. Before starting we have to understand that to work with different iFrames on a page we have to switch between these iFrames.



To Switch between Frames we have to use the driver's switchTo().frame command.

We can use the **switchTo().frame() in four ways**:

1. Switch to Frames by Index:

Index is one of the attributes for frame handling in Selenium through which we can switch to it. The index of frame starts from 0. Pass the frame index and driver will switch to that frame. NoSuchFrameException is thrown if the frame is not found.

Suppose if there are 100 frames in page, we can switch to frame in Selenium by using index.

driver.switchTo().frame(0); driver.switchTo().frame(1);

In the sample page, we have two IFrames, index of iFrame starts from 0. So there are two iFrames on the page with index 0 and 1. Index 0 will be the iFrame which exists earlier in the HTML page. to switch to 0th iframe we can simple write driver.switchTo().frame(0). Here is the sample code:

```
public static void main(String[] args) throws InterruptedException {
        WebDriver driver = new FirefoxDriver();
        driver.get("https://demoqa.com/frames");

        //Switch by Index
        driver.switchTo().frame(0);
        driver.quit();
}
```

2. Switch to Frames by Name:

This method uses the frame name as defined by the developer as the parameter. The frame name is considered a String and is enclosed in quotes. Pass the frame element Name or ID and driver will switch to that frame. NoSuchFrameException is thrown if the frame is not found.

Syntax: switchTo().frame(frameName)

Now if you take a look at the HTML code of iFrame you will find that it has Name attribute. Name attribute has a value iframe1. We can switch to the iFrame using the name by using the command driver.switchTo().frame("iframe1"). Here is the sample code.

```
WebDriver driver = new FirefoxDriver();
driver.get("https://demoqa.com/frames");

//Switch by frame name
driver.switchTo().frame("iframe1");
driver.quit();
```

3. Switch to Frame by ID:

This method uses the frame id as the parameter. Similar to the name attribute in the iFrame tag we also have the ID attribute. We can use that also to switch to the frame. All we have to do is pass the id to the switchTo command like this driver.SwitchTo().frame("IF1"). Here is the sample code:

Syntax: switchTo().frame(frameID)

```
WebDriver driver = new FirefoxDriver();
driver.get("https://demoqa.com/frames");

//Switch by frame ID
driver.switchTo().frame("IF1");
driver.quit();
```

4. Switch to Frame by WebElement:

This method uses the webelement as the parameter. Pass the frame web element and driver will switch to that frame. NoSuchFrameException is thrown if the frame is not found. StaleElementReferenceException if the frame is no longer active.

Syntax: switchTo().frame(WebElement)

Now we can switch to an iFrame by simply passing the iFrame WebElement to the driver.switchTo().frame() command. First, find the iFrame element using any of the locator strategies and then passing it to switchTo command. Here is the sample code:

```
WebDriver driver = new FirefoxDriver();
driver.get("https://demoqa.com/frames");
//First find the element using any of locator stratedgy
WebElement iframeElement = driver.findElement(By.id("IF1"));

//now use the switch command
driver.switchTo().frame(iframeElement);
driver.quit();
```

Switching back to the Main page from Frame:

There is one very important command that will help us to get back to the main page. The main page is the page in which two iFrames are embedded. Once you are done with all the tasks in a particular iFrame you can switch back to the main page using the switchTo().defaultContent(). This method is for switching to and fro in between frames and parent frames. The focus is shifted to the main page. Here is the sample code which switches the driver back to the main page.

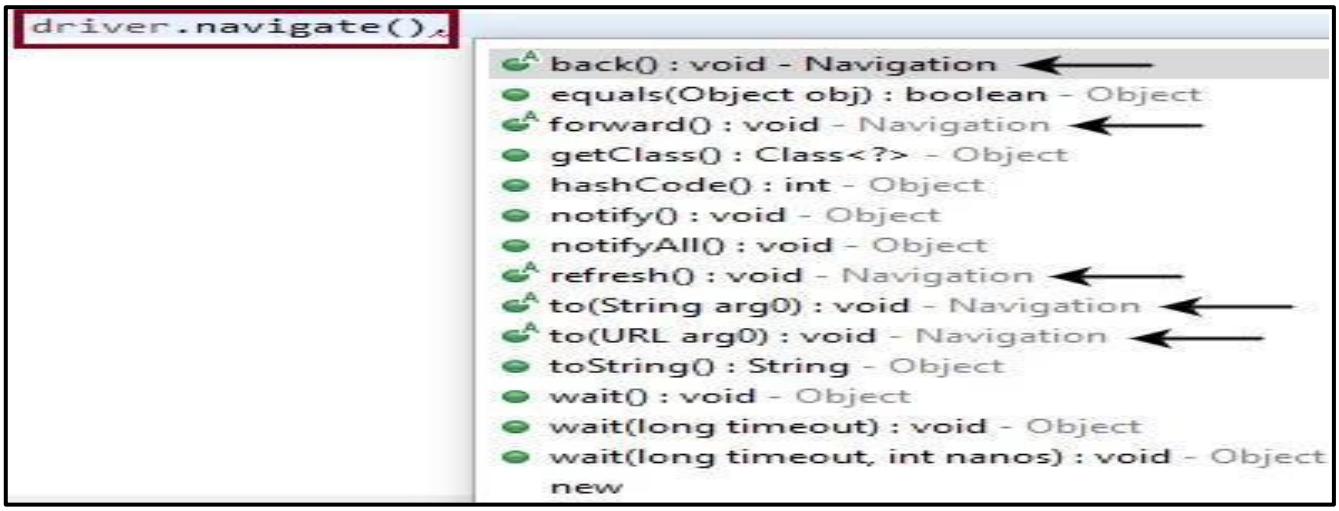
```
WebDriver driver = new FirefoxDriver();
driver.get("https://demoqa.com/frames");
//First find the element using any of locator stratedgy
WebElement iframeElement = driver.findElement(By.id("IF1"));

//now use the switch command
driver.switchTo().frame(0);

//Do all the required tasks in the frame 0
//Switch back to the main window
driver.switchTo().defaultContent();
driver.quit();
```

Method	Description
frame(WebElement element)	Selects a frame by WebElement
frame(String nameOrId)	Selects a frame by its name or ID
frame(int index)	Selects a frame by its (zero-based) index
defaultContent()	Selects either first frame or main document
parentFrame()	Change focus to the parent context

****

Browser Navigations in Selenium

During QA automation, there are instances when you need to go back to the previous page or move forward to the next page in browser history. WebDriver provides some basic browser navigation commands that allow the user to move backward or forward in the browser's history. It has methods to move back, forward as well as to refresh a page. We can access the navigation methods provided by WebDriver by typing driver.navigate() in the Eclipse panel. The first thing you will want to do after launching a browser is to open your website. To access the navigation's method, just type driver.navigate(). The IntelliSense feature of the eclipse will automatically display all the public methods of Navigate Interface shown in the below image.



WebDriver provides some basic Browser Navigation Commands that allows the browser to move backwards or forwards in the browser's history.

Browser Navigation commands for Selenium WebDriver:

1. Navigate To Command: To launch a new web browser window and navigate to the specified URL.

Method: to(String arg0) : void

In WebDriver, this method loads a new web page in the existing browser window. It accepts String as parameter and returns void. The respective command to load/navigate a new web page can be written as:

Command - driver.navigate().to(appUrl);

It does exactly the same thing as the driver.get(appUrl) method. Where appUrl is the website address to load. It is best to use a fully qualified URL.

driver.navigate().to("https://nallakrishna.in/RegLogin/Info.html ");

Note: The get command (driver.get(URL);) which lies in the browser commands section does the same function as the navigate command

2. Forward Command: This method does the same operation as clicking on the Forward Button of any browser. It neither accepts nor returns anything. Takes you forward by one Page on the browser's history. To navigate to the next web page with reference to the browser's history.

Command: driver.navigate().forward();

3. Back Command: This method does the same operation as clicking on the Back Button of any browser. It neither accepts nor returns anything. Takes you back by one page on the browser's history. Takes back to the previous web page with reference to the browser's history. This method enables the web browser to click on the back button in the existing browser window.

Command: driver.navigate().back();

4. Refresh Command: This method Refresh /reloads the current web page in the existing browser window. It neither accepts nor returns anything. Perform the same function as pressing F5 in the browser. To refresh the current web page thereby reloading all the web elements.

Command: driver.navigate().refresh();

Example for Selenium Navigation Commands:

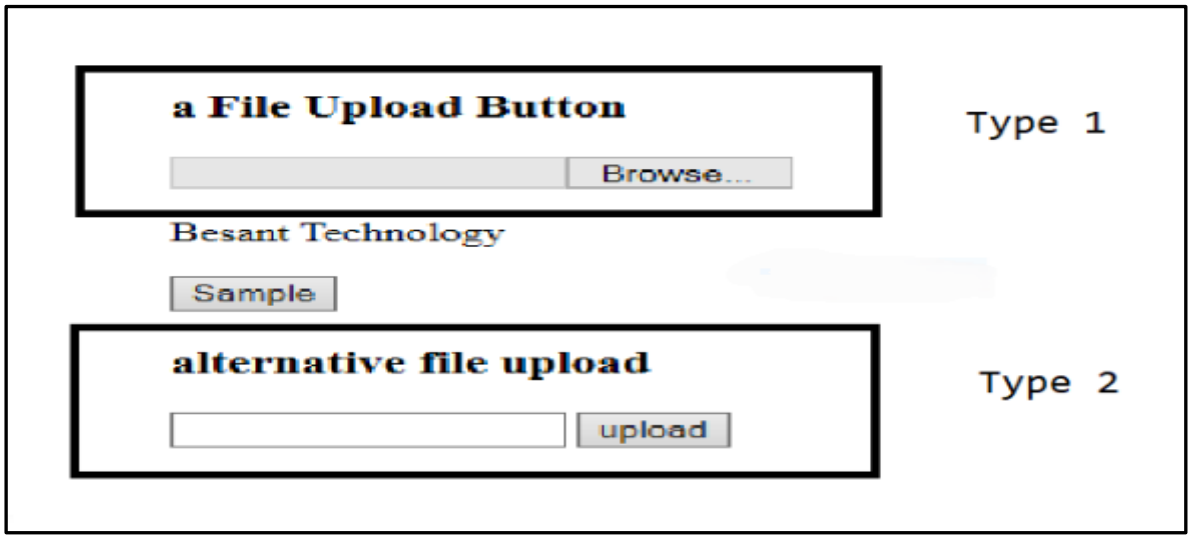
- ✓ Open DemoQA.com website
- ✓ Launch new Browser
- ✓ Click on Registration link using "driver.findElement(By.xpath(".//[@id='menu-item-374']/a")).click();
- ✓ Come back to Home page (Use 'Back' command)
- ✓ Again go back to Registration page (This time use 'Forward' command)
- ✓ Again come back to Home page (This time use 'To' command)
- ✓ Refresh the Browser (Use 'Refresh' command)
- ✓ Close the Browser

****

File Upload & Download in Selenium

While Selenium testing you may have come across a requirement where you need to either download or upload file in Selenium. Almost every web-application over the internet may have a feature for allowing users to either download or upload a file.

I. **Upload File:** When performing website testing, validating file upload functionality is crucial as users need to upload documents and images, such as in job portals, eCommerce websites, cloud storage systems, and others. When handling file uploads, Selenium offers extensive browser support, robust automation capabilities, and flexibility for efficiently handling different file types and dynamic interactions. Instead of interacting with the file upload dialog, it provides a way to upload files without opening the dialog box.



Methods to Handle File Upload in Selenium:

1) sendKeys() method

2) Robot class

3) AutoIt

As we all know that Selenium can be used to automate web applications. These web applications in Selenium use the underlying web object (locators) of the web app to determine the position on the web page and operate accordingly. Now in certain cases, we see that we need to handle windows authentication pop-up or any other windows pop-up while automating on the Windows operating system. Selenium cannot automate Windows actions but we can use sendKeys() method ,Robot Class and AutoIt in Java to accomplish the task.

1) Using sendKeys Method: The most basic way of uploading files in Selenium is using the sendKeys method. It is an inbuilt feature for file upload in Selenium. Uploading files in WebDriver is done by simply using the sendKeys() method on the file-select input field to enter the path to the file to be uploaded. A message is displayed after uploading a file that confirms if the file is being uploaded successfully.

```
WebElement upload_file = driver.findElement(By.xpath("//input[@id='file_up']"));
upload_file.sendKeys("C:/Users/krish/Desktop/upload.png");
```

For uploading files using this method, we first need to inspect the element or the button provided for file upload, then by using sendKeys, browse the path where the actual file to be uploaded is kept. Place the path along with the file name in sendKeys so that the program is navigated to the mentioned path to fetch the file. After this, click on the save or submit button and the file will be seen uploaded. At times, we also receive a message that the file is being uploaded successfully.

2) Using Robot Class Method: Selenium cannot automate Windows actions but we can use Robot Class in Java to accomplish the task. Robot Class in a separate class which is part of Java not a part of Selenium, Robot class is mainly created for automating Java Platform implementation. Robot class is a separate class in Java which will allow us to perform multiple tasks based on our requirement. It generally will throw AWT exception so we need to deal with it accordingly. Using Robot class, we can simulate keyboard event in Selenium. To use keyboard events you have to use to a method of Robot class: keyPress() and keyRelease().

```
// Create object of Robot class
 Robot robot = new Robot();
 // Press Enter
 robot.keyPress(KeyEvent.VK_ENTER);
// Release Enter
 robot.keyRelease(KeyEvent.VK_ENTER);
 // Press CTRL+V
 robot.keyPress(KeyEvent.VK_CONTROL);
 robot.keyPress(KeyEvent.VK_V);
// Release CTRL+V
 robot.keyRelease(KeyEvent.VK_CONTROL);
 robot.keyRelease(KeyEvent.VK_V);
```

3) Using AutoIt Method: AutoIt is an open-source test automation tool that automates native windows-related popups. AutoIt is a popular web testing framework. AutoIt v3 is a BASIC-like scripting computer language which is designed for automating the Microsoft Windows GUI [User Interface] and general scripting. It uses a combination of simulated keystrokes, mouse movement and window/control manipulation in order to automate tasks in a way not possible or reliable with other languages (e.g. VBScript and SendKeys). AutoIt is also very small, self-contained and will run on all versions of Windows out-of-the-box with no annoying "runtimes" required! AutoIt provides features to handle Windows dialogs, automate desktop applications, and interact with non-browser elements.

ControlFocus("Open","","Edit1") [ControlFocus() --> focus on the text box]
ControlSetText("Open","","Edit1","C:\Users\PrajjawalK\Music\check\visit.pdf") [ControlSetText() --> providing path of a file]
ControlClick("Open","","Button1") [ControlClick() --> clicking on open button]

- After writing above AutoIt script in Autoit editor and save it as "**UploadFile.au3**" [Filename. Extension] in your system. □ Once the file is saved.

Compile AutoIT script and generate .exe file

Tools-->Compile-->Select x64--> Compile --> generated .exe file **Use and integrate .exe file in selenium webdriver script:**

Ex: Runtime.getRuntime().exec("C://autoitfiles/fileupload.exe);

II. Download File:

One of the most common scenarios among internet users is downloading files off web pages. While Selenium doesn't support this functionality natively, there is an easy way to use Selenium to download a file. The downloading process or approach is different in different browsers – such as Firefox and Chrome. So, if a tester is using Selenium Webdriver to download files they need to have separate configurations for each browser.

```
File downloadedFile = new File("C:\\path\\to\\download\\directory\\filename.ext");
 downloadedFile.click();
```

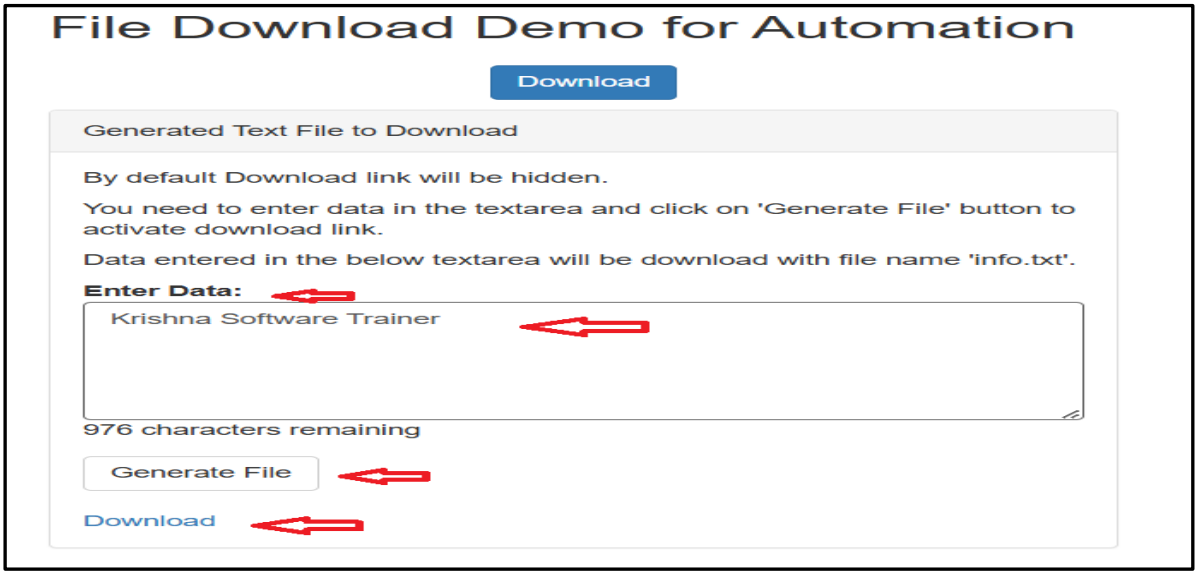
We will be automating the following **Test Scenario** to download files using JavaScript and Selenium:

Step 1: Go to the website: https://demo.automationtesting.in/FileDownload.html

Step 2: Enter data in the textarea.

Step 3: Click the "Generate File" button to generate the text file.

Step 4: Click the "Download" link to download the file.



```
//Action-1 Enter Text in TextBox
WebElement EnterTextFileData=driver.findElement(By.xpath("//textarea[@id=\"textbox\"]"));
EnterTextFileData.sendKeys("Krishna Software Trainer");
Thread.sleep(2000);

//Action-2 Click on Generate File Button
WebElement GenerateTextFile=driver.findElement(By.xpath("//button[@id='createTxt']"));
GenerateTextFile.click();
Thread.sleep(2000);

//Action-3 Click on Download Link
WebElement DownloadTextFile=driver.findElement(By.xpath("//a[@id='link-to-download']"));
DownloadTextFile.click();
```

****

Handle Multiple Windows in Selenium

A **Window** in any browser is the main webpage on which the user is landed after hitting a link/URL. The main window is also known as the parent window. All other windows that open within the parent window are called child windows. Such a window in Selenium is referred to as the parent window also known as the main window which opens when the Selenium WebDriver session is created and has all the focus of the WebDriver.

When you open any website, the main page where you will perform any operation is the **Parent Window**. This is the same webpage that will open when the Selenium automation script is executed. Now, all the other windows that open inside your main window are termed as **Child Windows**.

For a QA engineer, the art of working with multiple windows is critical, given that the most commonly found scenarios in modern web applications are pop-ups, authentication windows, or even new tabs being opened in response to user actions. These situations require the ability to switch between windows or tabs, perform operations on each, and return to the original window without disrupting the flow of automation. In cases not handled in the right way, test processes can become inconsistent and lead to failing tests or missing defects.

Normally, when a Selenium WebDriver instance is opened, only one window will be available by default. Whenever further windows or tabs are opened after that-for example, from clicking a link or from a pop-up-each one will have a window handle of its own. The latter allows Selenium to distinguish between each of the windows and switch between them by direct references inside the automation script.

Window Handle:

In Selenium, a window handle is a unique identifier given by Selenium to each window or tab opened during a test session. The window handle is basically a string value that serves to make it possible for Selenium to distinguish between different open windows or tabs and act on them separately. It is assumed that each browser will have a unique window handle. This window handle function helps to retrieve the handles of all windows.

A) get.windowhandle(): This method helps to get the window handle of the current window. When a website opens, we need to handle the main window i.e the parent window using driver.getWindowHandle( ); method. With this method, we get a unique ID of the current window which will identify it within this driver instance. This method will return the value of the String type. Get the handle of the parent window using the command:

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

B) get.windowhandles(): To handle all opened windows which are the child windows by web driver, we use driver.getWindowHandles( ); method. The windows store in a Set of String type and here we can see the transition from one window to another window in a web

application. Its return type is Set <String>. This method helps to get the handles of all the windows opened.

```
Set<String> allWindowHandles = driver.getWindowHandles();
```

C) set<string>: This method helps to set the window handles in the form of a string. These window handles are put into a set of strings, Set, from getWindowHandles(). All child windows are stored in a set of strings.

```
set<String> set= driver.get.windowhandles()
```

D) switch to: This method helps to switch between the windows. Clicking a link which opens in a new window will focus the new window or tab on screen, but WebDriver will not know which window the Operating System considers active. To work with the new window, you will need to switch to it. When we have multiple windows in any web application, the approach may need to switch control among several windows i.e from one window to another to perform any action and we can achieve this by using switchto(); method.

```
driver.switchTo().window(ChildWindow);
```

Method	Description
getWindowHandle()	Gets the window handle of the current window
getWindowHandles()	Gets the window handle of all current windows
switchTo().window()	Switch focus between windows

Users can also use the following methods to manage windows in Selenium:

A. Get Window Size: You can fetch the size of the browser window in pixels using the following methods. You can use driver.manage().window().getSize() to obtain the dimensions of the browser window, which is useful for responsive design testing.

```
// Access each dimension individually
int width = driver.manage().window().getSize().getWidth();
int height = driver.manage().window().getSize().getHeight();
```

```
// Or store the dimensions and query them later
Dimension size = driver.manage().window().getSize();
int storedWidth = size.getWidth();
int storedHeight = size.getHeight();
```

B. Set Window Size: To set the window size, use the setSize method. Use driver.manage().window().setSize(new Dimension(width, height)) to set the window size to specific dimensions, ensuring your web application's behavior under different viewport sizes.

```
// Restore the window and set the size to 1024×768 pixels
driver.manage().window().setSize(new Dimension(1024, 768));
```

```
Dimension newDimension = new Dimension(1024, 768);
driver.manage().window().setSize(newDimension);
```

C. Get Window Position: You can fetch the coordinates of the top-left corner of the browser window using these techniques. Use driver.manage().window().getPosition() to determine the exact location of the browser window's top-left corner on the screen.

```
// Access each dimension individually
int x = driver.manage().window().getPosition().getX();
int y = driver.manage().window().getPosition().getY();
```

```
// Or store the dimensions and query them later
Point position = driver.manage().window().getPosition();
int storedX = position.getX();
int storedY = position.getY();
```

D. Set Window Position: Move the window to the desired position on the screen. With driver.manage().window().setPosition(new Point(x, y)), you can move the browser window to a particular position on the screen for targeted testing. We can set window position to 50 points from left side and 200 points from top side using bellow given syntax.

```
// Move the window to the top-left corner of the primary monitor
driver.manage().window().setPosition(new Point(50, 200));
```

E. Maximize Window: Used to enlarge the window to fill the screen (cross-platform). This is the most common one in all the Selenium projects, driver.manage().window().maximize() used to maximize the browser window, providing a consistent, full-screen testing environment across different browsers.

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

F. Minimize Window: Used to minimize the window of the current browsing context. Use driver.manage().window().minimize() to simulate minimizing the window and test how your web application behaves when minimized.

```
driver.manage().window().minimize();
```

G. Fullscreen Window: We can make the window fullscreen, similar to pressing F11 in most of the browsers. Make use of driver.manage().window().fullscreen(), to enter fullscreen mode, replicating the user's experience of pressing F11 in most browsers during testing.

```
driver.manage().window().fullscreen();
```

H. New Window: Opening an URL in a new window or new tab is one of the features provided by Selenium 4. In Selenium 3, we can achieve this by performing switch operation using the WindowHandle method. In Selenium 4, this is going to change by using the new API newWindow. In Selenium 3, we needed to create a new WebDriver object and then switch to a new tab or window. But with Selenium 4, we can avoid creating new WebDriver objects and directly switch to the required window. We have a new API called newWindow in Selenium 4. This API creates a new tab/window and automatically switches to it. You can open new window & tab using "newWindow" command.

Creating a new window in Selenium 4 and switching to it:

```
driver.switchTo().newWindow(WindowType.WINDOW);
```

Opening a New Window:

```
driver.get("https://www.selenium.dev/");
// A new window is opened and switches to it
driver.switchTo().newWindow(WindowType.WINDOW);
// Loads FaceBook website in the newly opened window
driver.get("https://www.facebook.com");
```

I. New Tab: Sometimes, we have to open a URL or link in a new tab but in the same window. This post will look at how we can open a new URL or a link in the same tab using Selenium Webdriver in Java. You can open new window & tab using "newWindow" command.

Here's how you can create a new tab in Selenium 4 and switch to it:

```
driver.switchTo().newWindow(WindowType.TAB);
```

Opening a New Tab:

```
driver.get("https://www.selenium.dev/");
// A new tab is opened and switches to it
driver.switchTo().newWindow(WindowType.TAB);
// Loads FaceBook website in the newly opened window
driver.get("https://www.facebook.com");
```

Close a Browser in Selenium:

driver.close() and driver.quit() are two methods for closing a browser session in Selenium WebDriver. It is necessary to know when to use each method in a test script.

J. driver.close(): close() is a webdriver command that closes the browser window currently in focus. During the automation process, if there is more than one browser window opened, then the close() command will close only the current browser window, which is having focus at that time. The remaining browser windows will not be closed. This method is used when you want to close the current existing window you are working on. Please note driver.close()

will not terminate the browser instance. driver.close() closes only the current window on which Selenium is running automated tests. The WebDriver session, however, remains active. The following code can be used to close the current browser window:

```
driver.close(); //closes the browser
```

K. driver.quit(): This method is used when you want to terminate the webdriver instance window you are working on. Please note driver.quit() will terminate the browser instance. driver.quit() method closes all browser windows and ends the WebDriver session. quit() is a webdriver command which calls the driver.dispose method, which in turn closes all the browser windows and terminates the WebDriver session. If we do not use quit() at the end of the program, the WebDriver session will not be closed properly, and the files will not be cleared off memory. This may result in memory leak errors.

```
driver.quit(); // terminates driver session and closes all windows
```

Note: If the Automation process opens only a single browser window, the close() and quit() commands work similarly. Both will differ in their functionality when there is more than one browser window opened during Automation.

"These window management techniques help you to interact with browser windows effectively during your Selenium WebDriver tests. Use them as needed to ensure comprehensive test coverage and a seamless user experience."

****

Handle Alerts and Popups in Selenium

Handling popups and alerts is critical to testing web applications using Selenium. In Selenium, popups and alerts can be handled using the Alert interface, which allows you to switch to the popup window, accept or dismiss the alert, and retrieve its text. For handling popups in Selenium, you need to identify the type of popup, switch to the popup window, interact with its elements, and switch back to the parent window. Handling popups and alerts effectively in Selenium ensures that your tests run smoothly and your web application functions correctly.

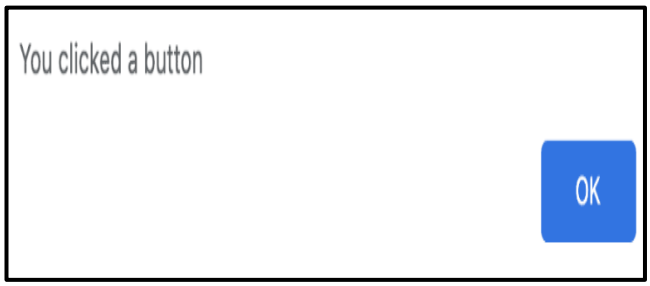
Alert: An Alert is nothing but a small message/notification box that appears on the screen to give some kind of information or asks for permission or give a warning for a potentially damaging operation or permission to perform that operation. It may be used for warning purposes as well. Sometimes, the user can enter a few details in the alert box as well.
Imagine filling out a form online and accidentally missing some information. You only know if you made a mistake if the website tells you somehow, like with a pop-up message. How do you know this unless there is an alert or a pop-up window to notify you?

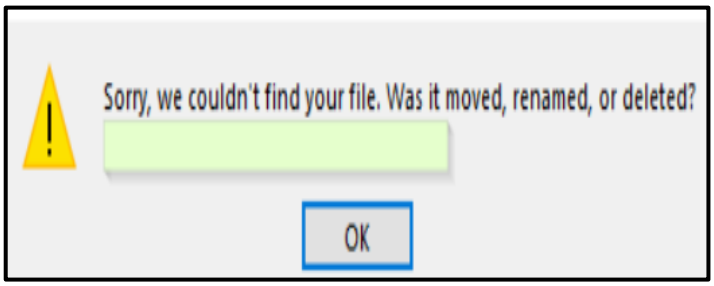
Pop-ups: Pop-ups are separate windows on top of the main window, often used for displaying advertisements, login prompts, or additional information. Selenium WebDriver can switch between multiple windows and handle pop-ups with built-in methods.

There are **3** types of Alerts in Selenium, described as follows:

1. Simple Alert	2. Prompt Alert	3. Confirmation Alert

1. Simple Alert: These alerts are just informational alerts. The simple alert in selenium shows some information or warning on the window and have an OK button on them. Users can click on the OK button after reading the message displayed on the alert box. This alert is used to notify a simple warning message with an 'OK' button, as shown in the below snapshot.





2. Confirmation Alert: The confirmation alert asks for the permission to do some type of operations. This alert is basically used for the confirmation of some tasks. These alerts get some confirmation from the user in the form of accepting or dismissing the message box. They are different from prompt alerts in a way that the user cannot enter anything as there is no text-box available. Users can only read the message and provide the inputs by pressing the OK/Cancel button. For Example: Do you wish to continue a particular task? Yes or No? The snapshot below depicts the same.

3. Prompt Alert: In Prompt alerts, some input requirement is there from the user in the form of text needs to enter in the alert box. This alert will ask the user to input the required information to complete the task. A prompt alert box is displayed like below, where the user can enter his/her username and press the OK button or Cancel the alert box without entering any details.

Handle Alerts in Selenium:

whenever an alert/popup appears, it opens up a new window. So, for handling the Alerts using Selenium WebDriver, the focus needs to be shifted to the child windows opened by the Alerts. To switch the control from the parent window to the Alert window, the Selenium WebDriver provides the following command:

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

Once we switch the control from the main browser window to the alert window, we can use the methods provided by Alert Interface to perform various required actions. For example, accepting the alert, dismissing the alert, getting the text from the alert window, writing some text on the alert window, and so on.

The following 4 methods are available to handle alerts in Selenium WebDriver:

1. void accept(): You can use the accept() method to accept an alert or confirm the dialog box. This method is used to accept or confirm an alert box. For example, if an alert box is displayed with the message "Are you sure you want to save changes?" the accept() method can be used to confirm the action. The void accept method is used to click on the 'OK' button of the alert.

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

```
// Switch to the alert box
Alert alert = driver.switchTo().alert();
// Click the OK button to accept the alert box
alert.accept();
```

2. void dismiss(): You can use the dismiss() method to dismiss an alert or confirm the dialog box. This method is used to dismiss or cancel an alert box. For example, if an alert box is displayed with the message "Are you sure you want to delete this item?" the dismiss() method can be used to cancel the action. The void dismiss method is used to click on the 'Cancel' button of the alert.

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

```
// Switch to the alert box
Alert alert = driver.switchTo().alert();
// Click the Cancel button to dismiss the alert box
alert.dismiss();
```

3. String getText(): This method is used to get the text displayed in an alert box. For example, if an alert box displays "Please enter your name", the getText() method can retrieve the message. This method captures the message from the alert box.

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

```
// Switch to the alert box
Alert alert = driver.switchTo().alert();
// Get the text displayed in the alert box
String alertText = alert.getText();
```

4. void sendKeys(String stringToSend): This method is used to send text input to an alert box. For example, if an alert box is displayed with the message "Please enter your email", the sendKeys() method can be used to input the email address. It is used to send some data to the prompt alert.

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

```
// Switch to the alert box
Alert alert = driver.switchTo().alert();
// Enter text in the alert box
alert.sendKeys("example@email.com");
```

The following code snippet demonstrates how to handle an alert in Selenium:

```
// Switch to the alert box
Alert alert = driver.switchTo().alert();

// Get the text displayed in the alert box
String alertText = alert.getText();

// Click the OK button to accept the alert box
alert.accept();

// Enter text in the alert box
alert.sendKeys("example@email.com");

// Click the Cancel button to dismiss the alert box
alert.dismiss();
```

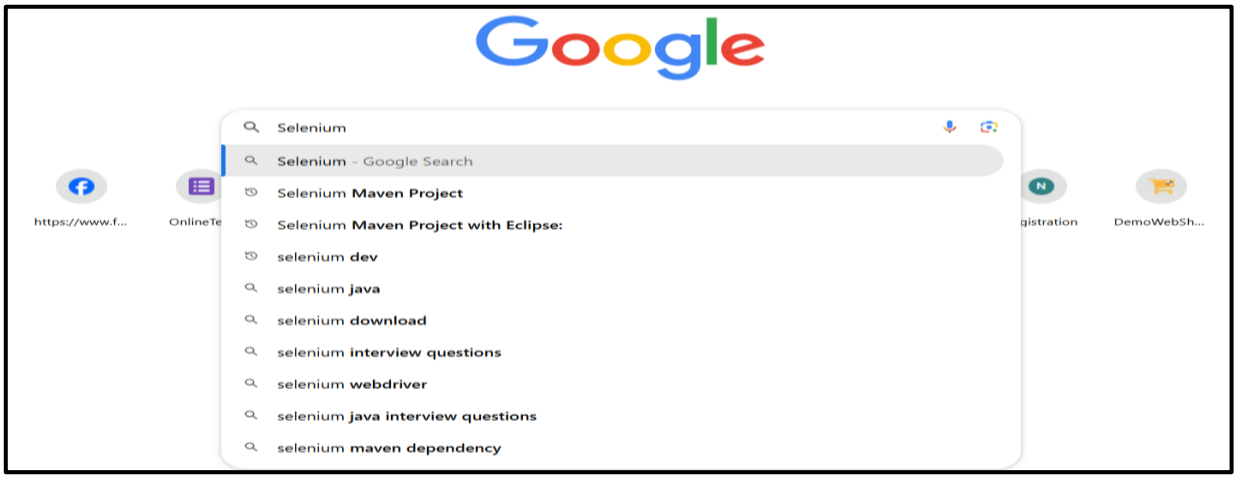
Method	Description
accept()	Accepts the alert
dismiss()	Dismisses / cancels the alert
getText()	Gets text from the alert
sendKeys()	Types text into the alert

****

Auto Suggestions and Auto Complete in Selenium

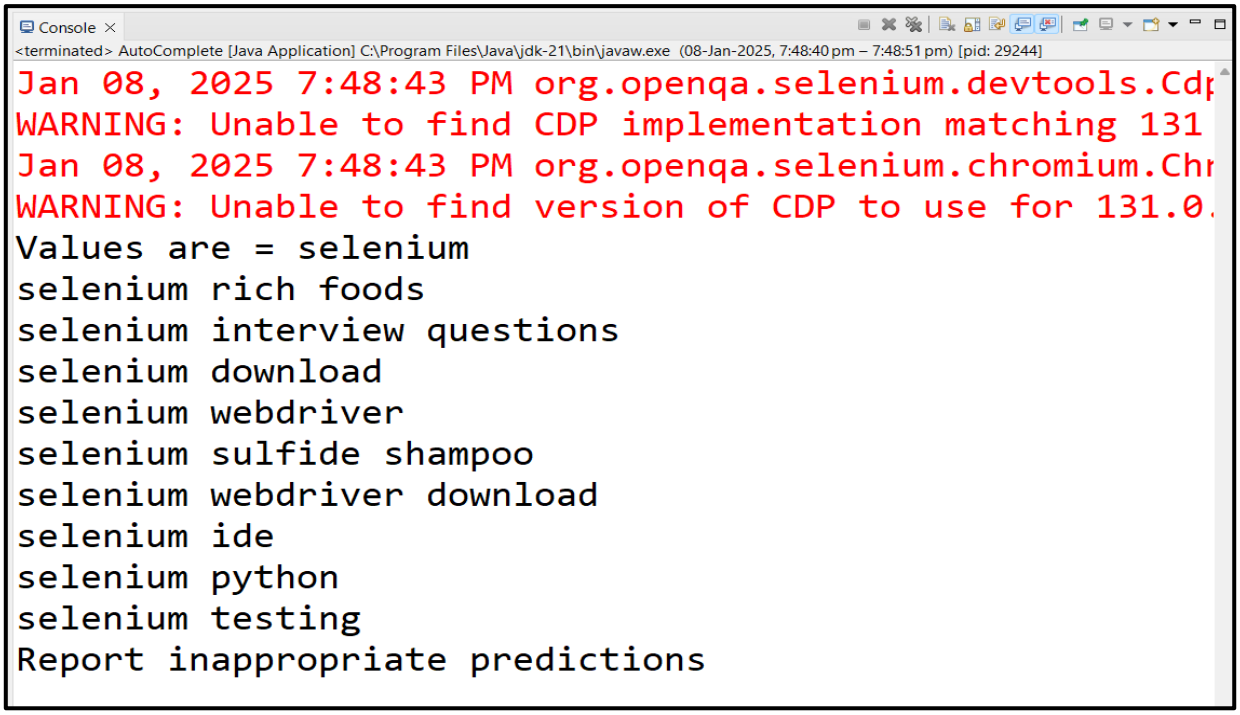
Auto Suggestion or Autocomplete allows the browser to predict the value. When a user starts to type in a field, the browser should display options to fill in the field, based on earlier typed values. For e.g. in Google Search box whenever the user enters any value google automatically tries to suggest to him the next text based on the value provided earlier. These suggestions aim to assist users in finding relevant information or completing their input more efficiently. These are implemented as a drop-down list with a list of possible suggestions as the user proceeds to input. This helps the user select the desired term from the list without entering it completely. The auto complete works with the following input types: text, search, url,email and datepickers.

In this example, we will try to automate Google Search box functionality in which I will try to search “selenium” and based on this input provided it will automatically search “Selenium-web driver” and click on the Search button.




```
package com.qa.practice;
import java.util.List;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
public class AutoComplete {
public static void main(String[] args) throws InterruptedException {
System.setProperty("webdriver.chrome.driver",
"C:\\Automation\\chromedriver_win32_2\\chromedriver.exe");
WebDriver driver=new ChromeDriver();
driver.get("https://www.google.co.in/");
driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
driver.findElement(By.name("q")).sendKeys("selenium");
```

```
List<WebElement> autoSuggest =
driver.findElements(By.className("UUbT9"));
Thread.sleep(3000);
// print the auto suggest
for (WebElement a : autoSuggest)     {
System.out.println("Values are = " + a.getText());
if (a.getText().equalsIgnoreCase("selenium webdriver"));
a.click();
Thread.sleep(3000);
break;     }
driver.close();        }  }
```



****

Web Tables in Selenium

Web tables, also known as HTML tables, are a widely used format for displaying data on web pages. They allow for a structured representation of information in rows and columns, making it easy to read and manipulate data. Selenium allows users to interact with these tables by locating elements based on their attributes, such as tags, classes, or IDs, enabling actions like extracting data, clicking on links within the table, or validating content.

A table consists of rows and columns. The table created for a web page is called a web table. Table is a kind of HTML data which is displayed with the help of <table> tag in conjunction with the <tr> and <td> tags. Below are some of the important tags associated with a web table:

< table > – Defines an HTML table

< th > – Contains header information in a table

< tr > – Defines a row in a table

< td > – Defines a column in a table

Example of writing a web table using HTML:

```
<table name="WebTableData">
  <tr>
    <th>Student Name</th>
    <th>Course Name</th>
    <th>Trainer Name</th>
  </tr>
  <tr>
    <td>Mounika</td>
    <td>Software Testing</td>
    <td>N.Krishna</td>
  </tr>
  <tr>
    <td>NagaRaju</td>
    <td>FullStack Python</td>
    <td>Srinivas Rao</td>
  </tr>
    <tr>
    <td>Arun</td>
    <td>FullStack Java</td>
    <td>N.Krishna</td>
  </tr>
</table>
```

Student Name	Course Name	Trainer Name
Mounika	Software Testing	N.Krishna
NagaRaju	FullStack Python	Srinivas Rao
Arun	FullStack Java	N.Krishna

In Selenium, the two main types of WebTables are:

Static WebTable: Static WebTables have a fixed structure and content that doesn't change after the page loads. The data is predefined, allowing for straightforward extraction and interaction since the number of rows and columns remains constant. Testing involves simply locating elements and retrieving values without concern for changes in the table's structure.

Dynamic WebTable: Dynamic WebTables, on the other hand, can change based on user interactions or data updates, such as sorting, filtering, or pagination. This type requires more complex handling because the number of rows, columns, or even the data within them can vary. Selenium scripts need to account for these changes, often involving methods to wait for elements to load or to interact with controls that modify the table's content.

Handling WebTables in Selenium: Below are the steps to handle WebTables in Selenium:

Step#1: Locate the table element: You can use Selenium's built-in methods, such as findElement(By.id()), findElement(By.cssSelector()), to locate the table element on the web page.

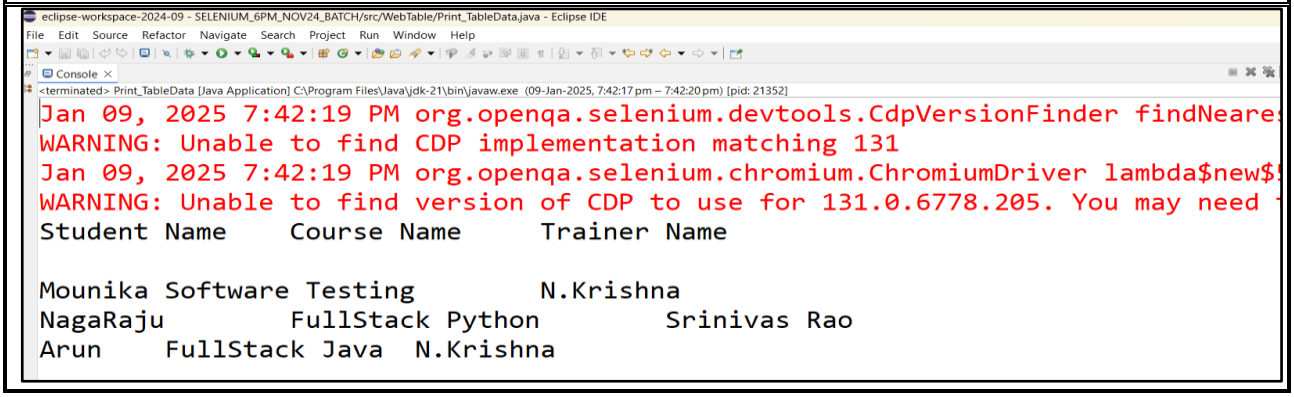
Step#2: Get the rows and columns of the table: You can use the findElement(By.cssSelector()) to select the rows and columns directly depending on the implementation of the table on the website.

Step#3: Perform actions on the elements within the table: Once you have the rows and columns of the table, you can perform actions on the elements within them using Selenium's WebDriver API. For example, you can retrieve the text of a specific cell, click on a button within a cell, or clear the contents of an input text field.

Printing Content of a Web Table In Selenium

```
package WebTable;
import java.util.List;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
public class Print_TableData     {
public static void main(String[] args)          {
System.setProperty("webdriver.chrome.driver","D:\\SELENIUM NOV-24
BATCH\\chromedriver-win64\\chromedriver.exe");
WebDriver driver = new ChromeDriver(); //Create WebDriver instance
driver.manage().window().maximize();
driver.get("https://www.nallakrishna.in/WebTable.html"); // Open the target URL
WebElement table =
driver.findElement(By.xpath("//table[@name=\"WebTableData\"]"));
 // Get all rows of the table
List<WebElement> rows = table.findElements(By.xpath(".//tr"));
```

```
for (int i = 0; i < rows.size(); i++)     { // Loop through each row
WebElement row = rows.get(i); // Get all cells (columns) in the current row
List<WebElement> cells = row.findElements(By.xpath(".//td"));
for (int j = 0; j < cells.size(); j++)   {     // Loop through each cell and print its text
WebElement cell = cells.get(j);
System.out.print(cell.getText() + "\t");  }    // Print cell data with a tab for spacing
System.out.println(); // Move to the next line after printing all columns in a row
}       }       }
```



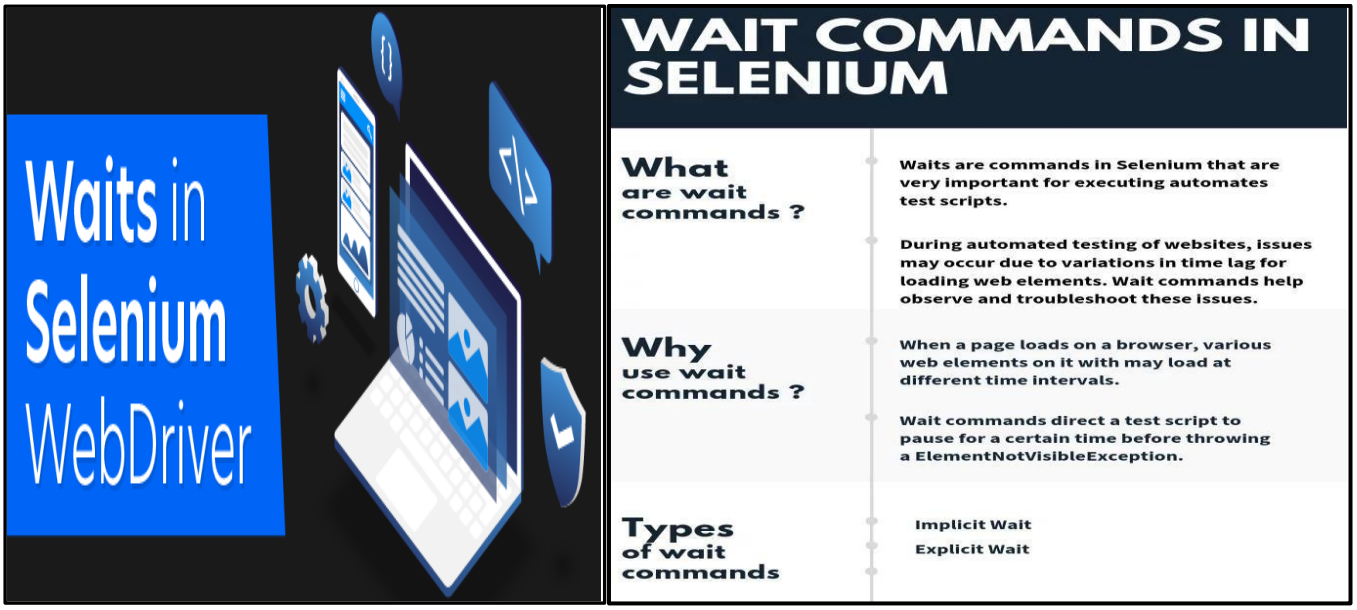
****

Implicit and Explicit Wait in Selenium

What is Wait in Selenium?

In Selenium, "Waits" play an important role in executing tests. In simple words, Selenium Wait is just a set of commands that wait for a specified period of time before executing test scripts on the elements. When to wait and how long to wait depends on the written script and type of wait used. You may be waiting for an element to load or become visible or you may want to wait till the complete page loads. While running Selenium tests, it is common for testers to get the message "Element Not Visible Exception". This appears when a particular web element with which WebDriver has to interact, is delayed in its loading. To prevent this Exception, Selenium Wait Commands must be used. Selenium Wait is therefore a preferred method when such scenarios are encountered or are expected on a web page. It helps prevent NoSuchElementFound exception.

In automation testing, wait commands direct test execution to pause for a certain length of time before moving onto the next step. This enables WebDriver to check if one or more web elements are present/visible/enriched/clickable, etc.

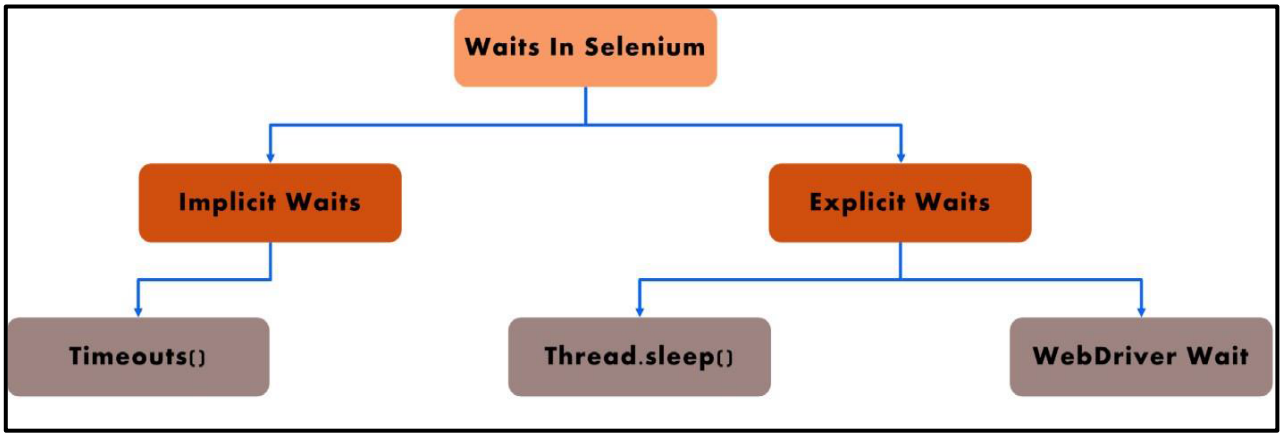


Why Do We Need Waits in Selenium:

Most of the web applications are developed using Ajax and Javascript. When a page is loaded by the browser the elements which we want to interact with may load at different time intervals. With this, it not only becomes difficult to identify the element but also if the element is not located, it will throw an "ElementNotVisibleException" exception. By using Waits, we can resolve this problem.

Types of Selenium Waits: Selenium supports two types of waits and they are as follows:

1. Implicit wait **2. Explicit wait**



1) Implicit wait in Selenium:

The Implicit Wait in Selenium is used to tell the web driver to wait for a certain amount of time before it throws a "No Such Element Exception". The default setting of Implicit wait is zero. Once you set the time, the web driver will wait for that particular amount of time before throwing an exception. This means that we can tell Selenium that we would like it to wait for a certain amount of time before throwing an exception that it cannot find the element on the page or the page is not loaded or the javascript execution is not finished. Also, important to note that once set, Implicit Wait stays in place for the entire duration for which the browser is open. An implicit wait is a condition-less wait command in Selenium. Since it is condition-less, it is applied to all the web elements on the web page. Selenium Web Driver has borrowed the idea of implicit waits from Watir.

We can apply implicit wait through three functions:

implicitlyWait()	pageLoadTimeout()	setScriptTimeout()

A) implicitlyWait Command:

During implicitlyWait, the WebDriver will poll the DOM for certain specified time units while trying to find any element. If the element is found earlier, the test executes at that point otherwise the WebDriver waits for the specified duration.

To add implicit waits in test scripts, import the following package.

import java.util.concurrent.TimeUnit;

Syntax: driver.manage().timeouts().implicitlyWait(TimeOut, TimeUnit.SECONDS);

Example: driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

Here, the tester has specified to wait for 10 seconds (through Duration.ofSeconds(10)) before moving ahead with the interaction with the element.

B) pageLoadTimeout Command:

As the name suggests, pageLoadTimeout command waits for the page to load completely for a specified number of seconds. The default value is 0 and a negative value means infinite wait.

Syntax: driver.manage().timeouts().pageLoadTimeout(TimeOut, TimeUnit.SECONDS);

Example: driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.SECONDS);

Here, the tester has specified to wait for 30 seconds before moving ahead to interact with the element. Therefore, WebDriver will wait for a maximum of 30 seconds.

For example, we can use network throttling and slow down the bandwidth to check page load time here. Using above code in the same scenario, we can know if our page can load on a 3G network or 2G network etc.

C) setScriptTimeout Command:

The setScriptTimeout command waits for the asynchronous parts of the web page to finish loading for a specified number of seconds.

Syntax: driver.manage().timeouts().setScriptTimeout(TimeOut, TimeUnit.SECONDS);

Example: driver.manage().timeouts().setScriptTimeout(15, TimeUnit.SECONDS);

Here, the tester has specified to wait for 15 seconds before moving ahead with the interaction with the element.

Note: The implicit wait is not advised to be mixed up with explicit wait in the test scripts. A test script with both the waits can produce unpredictable behavior due to erroneous(inaccurate) timeout durations.

Example Scenario:

You can proceed with the following exercises on Selenium Wait to make your basics stronger.

- → Launch a new browser (such as ChromeDriver).
- → Open URL "https://demoqa.com/dynamic-properties".
- → Maximize the window.
- → Find an element with id "visibleAfter". This element takes 5 secs to load which means this element will appear after 5 secs.
- → Use implicitlyWait to wait for that element.

Code: Add the above code into the test script. It sets an implicit wait after the instantiation of WebDriver instance variable.

```
package Waits;
import java.time.Duration;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class ImplicitWaitDEMOQA {
public static void main(String[] args) {
System.setProperty("webdriver.chrome.driver","C:\\Automation12pm
Class\\ChromeDriver\\chromedriver.exe" );
WebDriver driver = new ChromeDriver();
//driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
```

```
// launch Chrome and redirect it to the URL driver.get("https://demoqa.com/dynamic-properties"); driver.manage().window().maximize(); //This element will appear after 5 secs
driver.findElement(By.id("visibleAfter")).click();
//close browser
//driver.close();
} }
```

2) Explicit Wait in Selenium:

An explicit wait is a conditional wait strategy in Selenium in other words you wait until the condition you specified becomes true or the time duration has elapsed. Since explicit wait works with a condition, they help in synchronizing the browser, document object model, and the test execution script. Hence, the overall execution results are satisfactory and time-bound. The Explicit Wait in Selenium is used to tell the Web Driver to wait for certain conditions (Expected Conditions) or maximum time exceeded before throwing "ElementNotVisibleException" exception. It is an intelligent kind of wait, but it can be applied only for specified elements.

Conditions for Explicit wait in selenium web driver:

Condition 1: Suppose I have a web page which has some login form and after login, it takes a lot of time to load Account page or Home page. This page is dynamic it means sometimes it takes 10 seconds to load the homepage, sometimes it's 15 second and so on. In such situations, Explicit wait helps us to wait until a specific page is not present.

Condition 2: You are working on travel application and you have filled a web form and clicked on the submit button. Now you have to wait until the specific data is not displayed. In this case, again you can use Explicit wait in which you can give waits till specific or set of elements are not displayed.

Explicit wait provides the following conditions for usage:

alertIsPresent()	elementSelectionStateToBe()
elementToBeClickable()	elementToBeSelected()
frameToBeAvaliableAndSwitchToIt()	invisibilityOfTheElementLocated()
invisibilityOfElementWithText()	presenceOfAllElementsLocatedBy()
presenceOfElementLocated()	textToBePresentInElement()
textToBePresentInElementLocated()	textToBePresentInElementValue()
titleIs()	titleContains()
visibilityOf()	visibilityOfAllElements()
visibilityOfAllElementsLocatedBy()	visibilityOfAllElementsLocatedBy()

WebDriverWait Command in Selenium:

It is a concept of the dynamic wait which waits dynamically for specific conditions. It can be implemented by WebDriverWait class. WebDriverWait specifies the condition and time for which the WebDriver needs to wait. Practically, WebDriverWait and explicit wait go synonymously as their definitions and usage match perfectly. So if someone asks you to write some explicit wait scripts, it is safe to assume that the required scripts demand WebDriverWait.

To use Explicit Wait in test scripts, import the following packages into the script.

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

Then, Initialize A Wait Object using WebDriverWait Class.

Syntax: WebDriverWait wait=new WebDriverWait(WebDriveReference,TimeOut);

Example: WebDriverWait wait = new WebDriverWait(driver,30);

driver.until(ExpectedConditions.VisibilityofElementLocated(By.xpath(""//button[@value ='Save Changes']"")));

The explicit wait command helps us when we need to wait for a specific event before moving to the next step.

In explicit Wait, we must define a condition and the maximum time to wait.

Once the condition is fulfilled, Selenium WebDriver moves to the next step.

If the condition is unmet, it will throw a 'TimeoutException'.

Following is the expected condition that can be used for waiting:

Condition	Description
alertIsPresent()	Check if an alert element is present on the screen or not.
elementSelectionStateToBe()	This can be used to check if the element is selected or not by passing parameter
elementToBeClickable()	This used to instruct a command to wait until the element is clickable by the locator.
elementToBeSelected()	This check if an element is selected.
frameToBeAvaliableAndSwitchToIt()	It checks if we can switch to the given frame.
invisibilityOfTheElementLocated()	Checks for if the element is either invisible or not present on the dom.
invisibilityOfElementWithText()	Checks for if the element with text is either invisible or not present on the dom.
presenceOfAllElementsLocatedBy()	Checks for availability of the all elements present on the webpage with given locator.
presenceOfElementLocated()	It is the expectation for checking that an element is present on the DOM of a page.
textToBePresentInElement()	Checks for presence of text in the element.

textToBePresentInElementLocated()	Checks for the given text in an element located by a locator.
textToBePresentInElementValue()	Check if the text appears in given element value attribute.
titleIs()	It checks the title of the webpage.
titleContains()	It checks for a case sensitive substring in the title of a webpage.
visibilityOf()	Whether an element that is present on the DOM is visible or not.
visibilityOfAllElements()	Check for visibility of all elements in the webpage that matches a given locator.
visibilityOfAllElementsLocatedBy()	Checks for visibility of all elements. Height and width should be greater than 0.
visibilityOfElementLocated()	Check for visibility of an element.it should be visible as well as height and width should be greater than 0.

1. alertIsPresent()	9. textToBePresentInElementLocated()
2. elementSelectionStateToBe()	10. textToBePresentInElementValue()
3. elementToBeClickable()	11. titleIs()
4. invisibilityOfTheElementLocated()	12. titleContains()
5. invisibilityOfElementWithText()	13. visibilityOf()
6. presenceOfAllElementsLocatedBy()	14. visibilityOfAllElements()
7. presenceOfElementLocated()	15. visibilityOfAllElementsLocatedBy()
8. textToBePresentInElement()	16. visibilityOfElementLocated()

Implicit vs Explicit Waits:

Implicit Wait	Explicit Wait
It is a global wait and applied to all elements on the webpage.	It is a global wait and applied to all elements on the webpage.
Implicit Wait throws "NoSuchElementException" when the element does not appear even after the timeout.	Explicit Wait throws "TimeoutException" if the condition is not met in the specified time.
Implicit Wait waits before the element appears.	The explicit Wait can wait for many different conditions, such as the clickability of an element, etc.
Syntax:	Syntax:

driver.implicitly_wait(2);	w = WebDriverWait(driver, 7) w.until(expected_conditions.presence_of_element_located((By.ID, "name"))),
It is applied to all elements globally. Selenium WebDriver waits before searching for each element.	It is applied to a particular element or a group of elements.
The driver is asked to wait for a specific amount of time for the element to be available on the DOM of the page.	The driver is asked to wait for a specific amount of time for the element to be available on the DOM of the page.
It is simple and easy to implement.	It is simple and easy to implement.

❖ **Fluent Wait in Selenium:**

The Fluent Wait in Selenium is used to define maximum time for the web driver to wait for a condition, as well as the frequency with which we want to check the condition before throwing an "ElementNotVisibleException" exception. It checks for the web element at regular intervals until the object is found or timeout happens. The fluent wait is similar to explicit wait in Selenium with one additional argument of frequency (also known as polling time). The frequency number tells the WebDriver to keep checking for the element at regular intervals and wait till the maximum of "Duration.ofSeconds". This saves execution time. If the element becomes available earlier, we can proceed with our execution and finish quickly.

Example:

```
Wait wait = new FluentWait(WebDriver reference)
.withTimeout(timeout, SECONDS)
.pollingEvery(timeout, SECONDS)
.ignoring(Exception.class);
```

❖ **PageLoadTimeout Command in Selenium Timeout:**

As the name suggests, pageLoadTimeout command waits for the page to load completely for a specified number of seconds. The default value is 0 and a negative value means infinite wait.

WebDriver driver = new FirefoxDriver();
driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.SECONDS);
driver.get("https://www.google.com/");

Here, the tester has specified to wait for 30 seconds before moving ahead to interact with the element. Therefore, WebDriver will wait for a maximum of 30 seconds. The pageLoadTimeout is a good option when you are looking to test the page load performance to be within limits. For example, we can use network throttling and slow down the

bandwidth to check page load time here. Using above code in the same scenario, we can know if our page can load on a 3G network or 2G network etc.

❖ **Thread.sleep():**

Thread.sleep() is a static Java method that suspends the code for a specific time. It pauses the execution and helps us to know what has happened during the pause. It accepts the time specified in milliseconds. This function is particularly helpful for debugging a website or web page. Using Thread.sleep() in Selenium ensures smooth execution of automated tests, preventing script failures. When you call Thread.sleep(), the current thread enters a blocked state for the duration of the sleep period. During this time, the thread doesn't execute any code and doesn't consume any CPU resources. After the specified time has elapsed, the thread scheduler moves the blocked thread back to the list of threads that are ready to execute. The blocked thread is now ready to execute again, and the CPU can execute it along with other ready threads. Thread.sleep() is used to add a delay between operations or to wait for an element to appear on the page. This delay can be useful in a variety of scenarios, such as: **Waiting for page loading, Waiting for element visibility, Waiting for element interaction.**

Syntax: Thread.sleep(1000); //Pauses test execution for specified time in milliseconds

Using Thread.sleep() can be useful in some scenarios where you need a simple and quick way to introduce a delay in your test flow. For example, you may want to wait for a pop-up window to appear, a page to refresh, or a file to download. By adding a Thread.sleep() statement, you can ensure that your test does not proceed until the expected condition is met. This can help you avoid errors or failures due to timing issues or synchronization problems.

Selenium Wait	Description
Thread.sleep()	(Not a Selenium Wait Method) Causes execution to sleep for a certain number of milliseconds
Page Load Timeout	Sets the wait time for a page to load before throwing an error
Script Timeout	Sets the wait time for JavaScript to execute before throwing an error
Implicit Wait	Determines the wait time to search for an element before throwing an exception
Explicit Wait	Pauses execution until time has expired or an expected condition is met via WebDriverWait class
Fluent Wait	A specialization of Explicit Wait that's extended by the WebDriverWait class

****

Actions and Action Class in Selenium

Selenium Webdriver API does not support the user's actions like Mouse hover, right-click, context-click, and double-click. That is where the Actions class came to use, the actions provided by this class are performed by an API called Advanced user interaction in selenium webdriver.

Most user interactions like clicking on a button, entering text in textbox can be done using the WebDriver Element Commands. For e.g. Commands like WebElement.click() and WebElement.sendKeys() are used to click on buttons and enter text in text boxes. Submitting a form can be done using the WebElement.submit() command. Even interactions with drop-down are enabled using the Select class which exposes commands like selectByValue (), deselectAll () to select and deselect options.

However, there are complex interactions like Drag-n-Drop and Double-click which cannot be done by simple WebElement commands. To handle those types of advance actions we have the Actions class in Selenium.

Actions Class: "Actions" in Selenium refers to a higher-level API provided by the Selenium WebDriver library. It enables performing complex user interactions, such as mouse movements, key press events, drag-and-drop operations, and more. Actions allow you to chain multiple actions together to create sequences of actions or perform composite actions. Actions class is an ability provided by Selenium for handling keyboard and mouse events. In Selenium WebDriver, handling these events includes operations such as drag and drop in Selenium, clicking on multiple elements with the control key, among others. The Actions class in Selenium WebDriver provides methods like moveToElement(), click(), sendKeys(), dragAndDrop(), etc. You can create an instance of the Actions class, build a sequence of actions using its methods, and then perform those actions on the web page using the perform() method.

Actions class is a collection of individual Action that you want to perform. Actions class is an ability provided by Selenium for handling keyboard and mouse events. In Selenium WebDriver, handling these events includes operations such as drag and drop in Selenium, clicking on multiple elements with the control key, among others. These operations are performed using the advanced user interactions API. It mainly consists of Actions that are needed while performing these operations. Action class present in the package,

"org.openqa.selenium.interactions package"

Syntax: Actions actions = new Actions(driver);

Ex: actions.moveToElement(element).click().perform();

actions.click(element).perform();

Action Class: The term "action class" in Selenium is a more generic term that refers to any class or interface used to represent a specific user action or interaction. An action class encapsulates a single action or behavior that can be performed on a web element.

For example, if you want to perform a click action on a button, you can use the Actions class (as mentioned above) and its click() method to represent that action. Similarly, if you want to perform a hover action, you can use the moveToElement() method from the Actions class to represent the hover action. So, an action class is a specific implementation or representation of a particular user action, while the "Actions" in Selenium is a higherlevel API that provides a set of methods to perform various user interactions. Action Class in Selenium is a builtin feature provided by the selenium for handling keyboard and mouse events. It includes various operations such as multiple events clicking by control key, drag and drop events and many more. These operations from the action class are performed using the advanced user interaction API in Selenium Webdriver. Did I mention Action Class, actually it is not a class but an Interface. It is only used to represent the single user interaction to perform the series of action items build by Actions class.

What is the difference between Actions Class and Action Class in Selenium?

With the above explanations of Actions Class & Action Class, we can now conclude that Actions is a class that is based on a builder design pattern. This is a user-facing API for emulating complex user gestures. Whereas Action is an Interface which represents a single user-interaction action. It contains one of the most widely used methods perform().

Actions Class	Action Class
Part of the Selenium WebDriver API	Part of the Actions class in Selenium
Provides basic user interactions like click, send keys, etc.	Provides advanced user interactions like drag-and-drop, double-click, etc.
Uses the WebDriver instance directly to perform actions	Uses the Actions class to chain multiple actions and perform them as a single action
Works on a single element at a time	Can work on multiple elements simultaneously
Cannot perform complex interactions like drag-and-drop, hover, etc.	Can perform complex interactions like drag-and-drop, hover, etc.
Syntax: driver.findElement(By.id("elementId")).click();	**Syntax:** new Actions(driver).moveToElement(element).click().build().perform();
Introduced in Selenium 2.0	Introduced in Selenium 3.0
Supports basic mouse and keyboard interactions	Provides advanced interactions like drag and drop, doubleclick, etc.
Can only perform actions on a single element at a time	Can chain multiple actions together to perform complex interactions

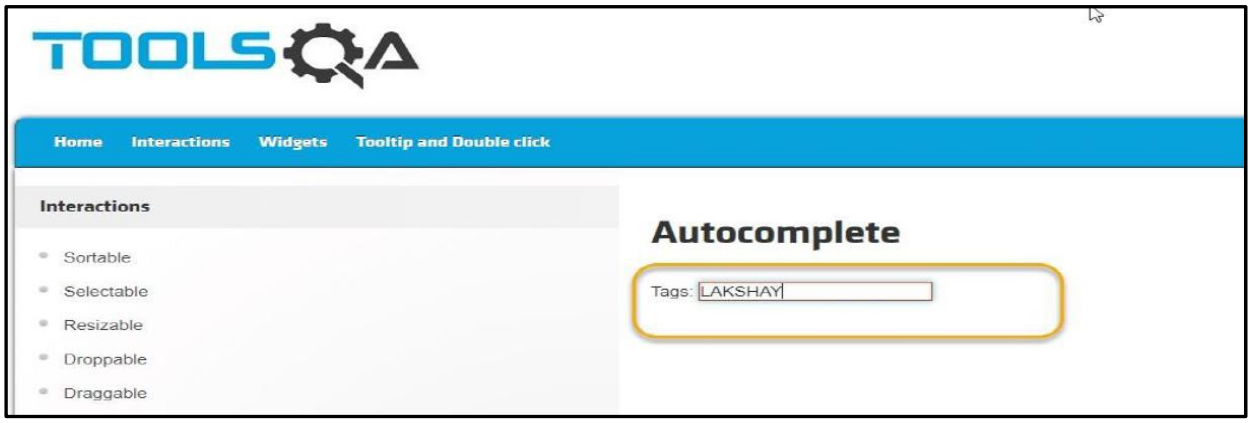
Doesn't provide support for handling rightclick context menus	Allows performing right-click actions and handling context menus

Actions and action class have different use cases depending on the complexity of the user interactions you need to perform. Actions are suitable for basic interactions on a single element, while the action class provides more flexibility and functionality for advanced interactions involving multiple elements.

How to use Actions class method in Selenium:

Let's understand the working of Actions class with a simple example:

Consider the scenario where it is required to enter Upper Case letters in the text box, let's take text box on ToolsQA's demo site http://demoqa.com/auto-complete/



Manually, it is done by pressing the Shift key and then typing the text which needs to be entered in Uppercase keeping Shift key pressed and then release the Shift key. In short Shift + Alphabet Key are pressed together.

1. Import Package:

Actions class & Action class reside in org.openqa.selenium.Interactions package of WebDriver API. To consume these, import their packages:

import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.interactions.Action;

2. Instantiate Actions Class:

Actions class object is needed to invoke to use its methods. So, let's instantiate Actions class, and as the Class signature says, it needs the WebDriver object to initiate its class.

Actions actions = new Actions(webdriver object);

3. Generate actions sequence:

Complex action is a sequence of multiple actions like in this case sequence of steps are:

Pressing Shift Key
Sending desired text
Releasing Shift key

For these actions, Actions class provides methods like:

Pressing **Shift Key** : Actions Class Method => keyDown
Sending desired **text** : Actions Class Method => sendKeys

Releasing Shift key : Actions Class Method => keyUp

The keyDown method performs a modifier key press after focusing on an element, whereas keyUp method releases a modifier key pressed. A modifier key is a key that modifies the action of another key when the two are pressed together like Shift, Control & Alt. Generate a sequence of these actions but these actions are performed on a webElement. So, let's find the web-element and generate the sequence:

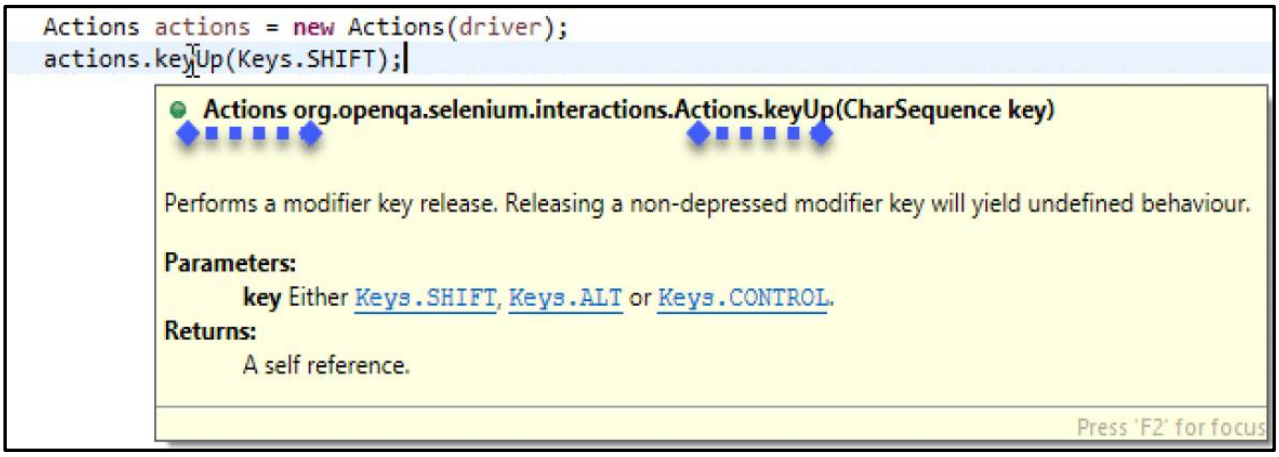
WebElement element = driver.findElement(By strategy to identify element);

actions.keyDown(element, Keys.SHIFT);

actions.sendKeys("TextToBeConvertAndSendInUpperCase");

actions.keyUp(Keys.SHIFT);

An important thing to note here is that, if you hover over any action class method, you will notice that it returns the Actions class object.



This is the beauty of the builder pattern. Which means that all actions can be clubbed together as below:

actions.keyDown(element,Keys.SHIFT).sendKeys("TextToBeConvertAndSendInUpperCase").keyUp(K eys.SHIFT);

4. Build the actions sequence:

Now, build this sequence using the build() method of Actions class and get the composite action. Build method generates a composite action containing all actions so far which are ready to be performed.

Action action = actions.build();

Notice that the build method returns the object type of Action. It is basically representing Composite Action which we built from a sequence of multiple actions. So, the second part of the Actions class description will get clear now, i.e. Actions class implements the builder pattern i.e. it Builds a CompositeAction containing all actions specified by the method calls.

5. Perform actions sequence:

And finally, perform the actions sequence using perform() method of Action Interface.

action.perform();

And this is done, once the execution passes this point, you will notice the action on the browser.

Same steps need to follow to leverage Actions class methods for performing different complex combinations of user gestures.

The Perform method is available in the Actions class as well. It means that it can be used directly as well without making the use of Action Interface like below:

actions.keyDown(element,Keys.SHIFT).sendKeys(TextToBeConvertAndSendInUpperCase).keyUp(Keys .SHIFT).perform();

Methods of Actions Class in Selenium:

Actions class is useful mainly for mouse and keyboard actions. In order to perform such actions, Selenium provides various methods.

There are a lot of methods in this class which can be categorized into two main categories:

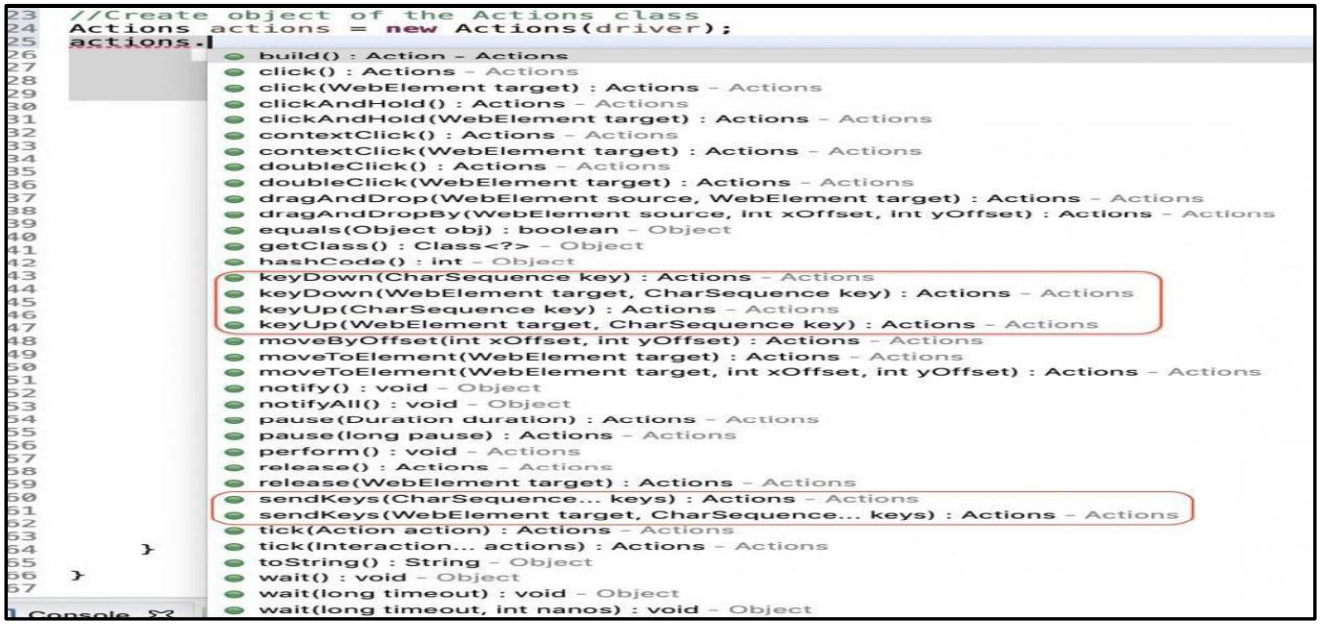
1.Keyboard Events **2.Mouse Events**

1. Keyboard Events in Selenium:

Selenium WebDriver provides a class named "Actions", which provides various methods that can help in automating and simulating the Keyboard and Mouse actions. The below figure shows the exhaustive list of methods offered by Selenium Web Driver, and the highlighted ones are the most used methods for simulating the Keyboard actions.

→ **Different Methods for performing Keyboard Events:**

- keyDown(modifier key): Performs a modifier key press.
- sendKeys(keys to send): Sends keys to the active web element.
- keyUp(modifier key): Performs a modifier key release.



A) keyDown(): This method simulates a keyboard action when a specific keyboard key needs to press. So, whenever you need to press a key and then perform specific other actions, we can use the keyDown() method to keep the key pressed. It accepts the modifier key as a parameter and performs a press action until specified to release the mentioned Key. And these standard modifier keys are Keys.SHIFT, Keys.Control, Keys.TAB etc

Syntax: keyDown(WebElement obj, Keys.modifierKey) or keyDown(Keys.modifierKey)

Ex: say a user has to type some characters in Capital. Then to simulate user behavior, where the user presses the SHIFT key and then presses the set of characters that need to type in Capital.

keyDown(CharSequence key): The following screenshot shows the syntactical details of this method: keyboard events keyDown method of Actions class in Selenium WebDriver.

This method presses the specified key on the currently focussed Web Element. This method generally presses the "Modifier keys" such as SHIFT, CTRL, etc. If you want to press the keyboard key on a specified web element, then that web element first needs to be focussed explicitly, and then this method needs to be invoked.

keyDown(WebElement element, CharSequence key): The following screenshot shows the syntactical details of this method:

Overloaded keyDown method of Actions class in Selenium WebDriver

This method first focusses on the web element, which has been passed as a parameter to the method and presses the mentioned key on that Web Element.

B) sendKeys(): This implementation of the sendKeys() method sends a sequence of characters/keys to a specific web element, which passes as the first parameter to the method. This method first focuses on the target web element and then performs the same action as sendKeys(CharSequence keys). This method sends a series of keystrokes to a given web element.

Ex: sendKeys(CharSequence... KeysToSend)

This method sends a sequence of keys to a currently focused web element, i.e., if we want to send specific characters to a web element, that element must be first focussed, then only the mentioned characters will go to that web element.

Ex: sendKeys(WebElement element, CharSequence... KeysToSend).

Ex: Consider a scenario in which a developer has created a website, and now the website needs to be tested for functionality. Here's where QAs come into the picture. They start writing test scripts using Selenium

WebDriver in order to replicate user actions on a browser. In some cases, QAs need to provide data in specific data input fields to validate functionality.

For example, to test a login page, the username and password fields require some data to be entered. Here's where QAs use the sendkeys() method to enter the field values.

The Selenium Sendkeys() method helps with field auto-completion in two simple steps:

Identifying the input fields using specific locators. For example, an email address, password field.

Entering values in input boxes explicitly using sendkeys() method.

C) keyUp(): It accepts modifier key as a parameter and performs key release action for a specified mentioned Key. **public Actions keyUp(Keys theKey)** : Performs a modifier key release (SHIFT,Keys.ALT or Keys.CONTROL) to Handle keyUp operation.

public Actions keyUp(WebElement element, Keys theKey) : performs a modifier key release after focusing on an element to perform keyUp operation.

Syntax: keyUp(WebElement obj, Keys.modifierKey) or keyUp(Keys.modifierKey)

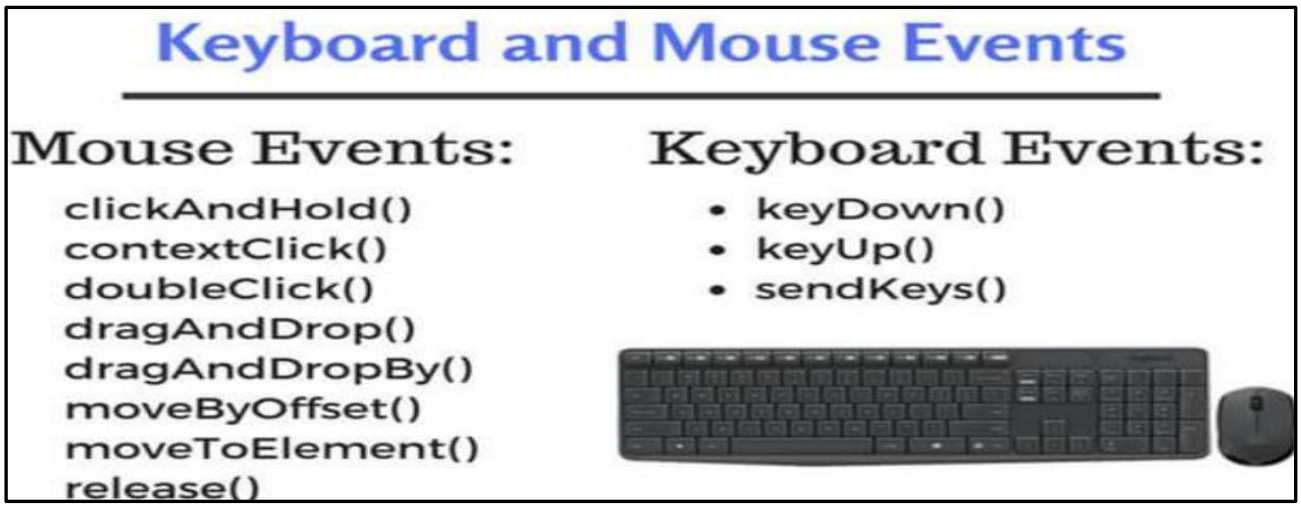
2. Mouse Events in Selenium:

Selenium Java has a variety of mouse actions that can be used to automate interactions with a website. Mouse actions in Selenium provide a mechanism for automating low-level elementary interactions such as mouse clicks, mouse hover, and mouse button actions. It also helps with automating complex low-level interactions such as mouse hover, drag and drop, click and hold, and more.

→ **Different Methods for performing Mouse Events:**

- click(): Clicks at the current mouse location.
- doubleClick(): Performs a double-click at the current mouse location.
- contextClick() : Performs a context-click at middle of the given element.
- clickAndHold(): Clicks (without releasing) in the middle of the given element.
- dragAndDrop(source, target): Click-and-hold at the location of the source element, moves to the location of the target element.
- dragAndDropBy(source, xOffset, yOffset): Click-and-hold at the location of the source element, moves by a given offset.
- moveByOffset(x-offset, y-offset): Moves the mouse from its current position (or 0,0) by the given offset.
- moveToElement(toElement): Moves the mouse to the middle of the element.
- release(): Releases the depressed left mouse button at the current mouse location.

There is a huge collection of methods available in Actions class. The below screenshot shows the list of methods available.



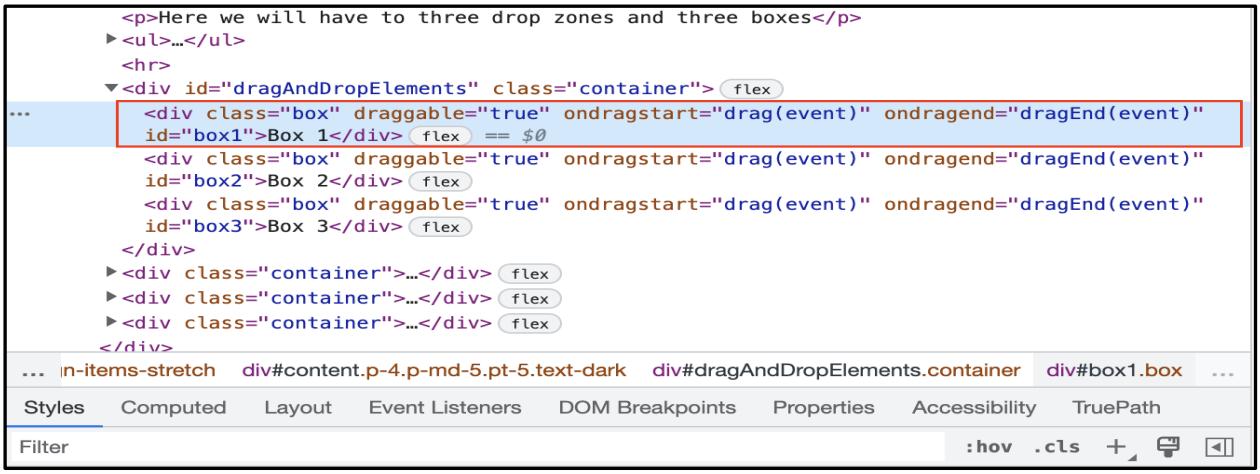
1. clickAndHold(): Click and hold is an action in Selenium in which we do left-click on an element and hold it without releasing the left button of the mouse. To perform this action in Selenium WebDriver, we will use clickAndHold() method of actions class. This method is generally useful in executing drag and drop operations. clickAndHold() method in Selenium helps simulate clicking and to hold down the mouse button on an element. It's part of the Actions class, which provides a way to perform complex user interactions with the web page.

Actions class has two overloaded implementations of the clickAndHold() methods:

A . clickAndHold(WebElement target) **B. clickAndHold()**

A. clickAndHold(WebElement target): It accepts a WebElement parameter, and the clickAndHold() method will move to that element and click (without releasing) in the middle of the element. In other words, it clicks and holds the mouse button on the specified element.

Let's now use the clickAndHold() method on the highlighted box in the below image.



First, we will have to find the element. So, let's inspect the highlighted box. We can see that the id attribute value is "box1". We can use this to find the element using the findElement() method.

WebElement Click_Hold = driver.findElement(By.id("box1"));

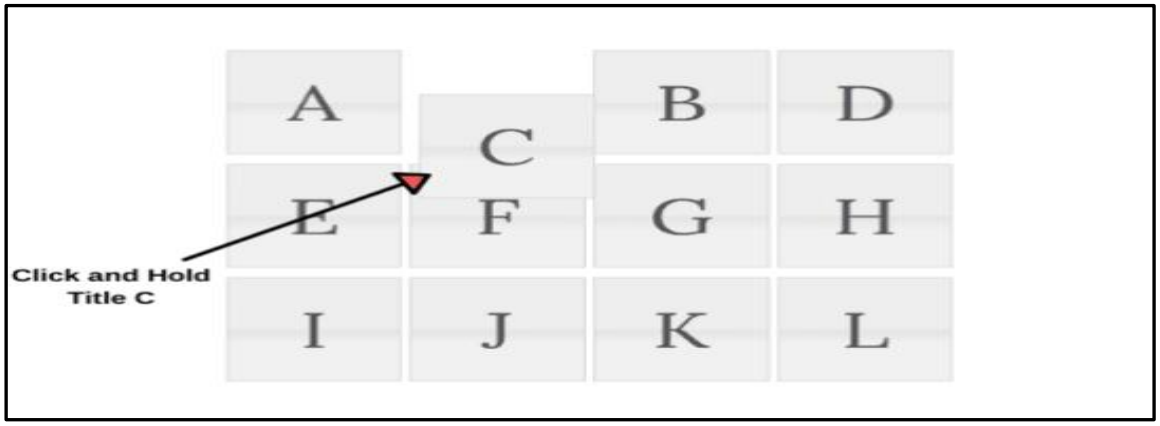
And then, we can use the click and hold the element using the code below.

Actions actions = new Actions(driver);

actions.clickAndHold(Click_Hold).build().perform();

B. clickAndHold(): The clickAndHold() method is a part of the Selenium WebDriver Actions class in Java. This method is used to perform a click-and-hold action on a web element. Typically, it is used in combination with other methods, such as moveToElement(), release(), and perform(), to simulate complex mouse interactions like drag-and-drop.

Ex: **action.clickAndHold().build().perform();**



2. contextClick() in Selenium [Right-click]: While automating a website for testing there is always required to perform some right-click or other user actions on the page. When a user performs a right click using the mouse on a particular element to perform some actions

is called a right click. Right click action in Selenium web driver can be done using Actions class. Right Click operation is also called Context Click in Selenium. Pre-defined method context click provided by Actions class is used to perform right click operation. Below is the code to demonstrate right click operation using Actions class.

Let's see how to use Action class methods to right click:

First, instantiate an Actions class:

Actions actions = new Actions(driver);

Now as seen above, the contextClick() method has argument WebElement to be passed. So, need to pass a WebElement object to the method. This WebElement should be the button or any web element on which we want to perform Right click. To find the element use the below command:

WebElement elementLocator = driver.findElement(By.id("ID"));

Now, when we have got the action class object and the element as well, just invoke perform() method for the right-click:

actions.contextClick(elementLocator).perform();

Let's see what happens internally when invoke the perform() method above:

Move to Element: contextClick() method first performs mouseMove to the middle of the element location. This function performs the right click at the middle of the web element.

Build: build() method is used to generate a composite action containing all actions. But if you observe, we have not invoked it in our above command. The build is executed in the perform method internally.

Perform: perform() method performs the actions we have specified. But before that, it internally invokes build() method first. After the build, the action is performed.

Example:

In this example, we are navigating to the URL "https://demoqa.com/buttons" and performing the Right click on the "Right click" button.

```
package programs;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.interactions.Actions;

public class RightClick
{
    public static void main(String[] args) throws InterruptedException {

        WebDriver driver;
        System.setProperty("webdriver.chrome.driver","C:\\Selenium 2023\\chromed
         driver= new ChromeDriver();
         driver.get("https://demoqa.com/buttons");
         driver.manage().window().maximize();
           WebElement element = driver.findElement(By.id("rightClickBtn"));
                Actions action = new Actions(driver);
                action.contextClick(element).perform();
                driver.close();
```

Explanation: Initially, we opened the browser and navigated to the URL

ChromeDriver driver = new ChromeDriver();

driver.manage().window().maximize(); driver.get("https://demoqa.com/buttons");

After that, we locate the web element where we have to perform the “Right Click”. Then, we initialize the Action class and performed “Right click” on the web element. Actions action=new Actions(driver); action.contextClick(element).perform();

Output: click is performed and the result will be displayed.

3. doubleClick(): Double-click is a frequently used user action. The most common use of double click happens in File Explorer E.g. In File Explorer, any Folder or File in a folder can be opened by double-clicking on it. Similarly, on any webpage, there might be some elements that require a double click to invoke an action on them. Similar to Right Click, in Selenium there is no WebDriver API command which has the capability to double click the webelement.

Hence, the Action class method doubleClick(WebElement) is required to be used to perform this user action.

Let's see how to use Actions class methods to double click:

First, let's instantiate an Actions class

Actions actions = new Actions(driver);

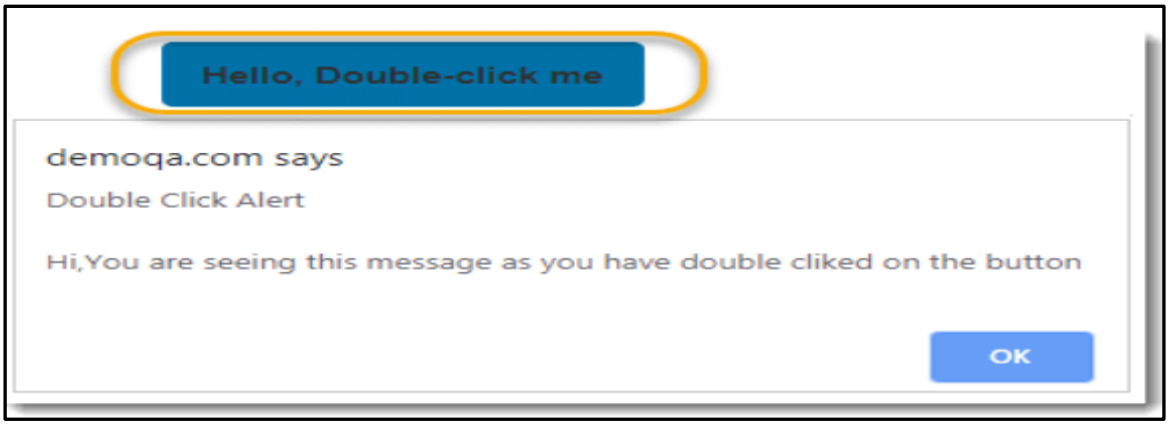
Now as seen above, doubleClick() method has argument WebElement to be passed. So pass WebElement object to the method on which need to perform Double click.

WebElement btnElement = driver.findElement(By.id("doubleClickBtn"));

Use any By strategy to locate the WebElement like find element by its id, name attribute, etc. Now, just invoke build and perform for our double click

actions.doubleClick(btnElement).perform();

doubleClick() method also follows the same process of Move to Element >> Build >> Perform which is same as for right-click.

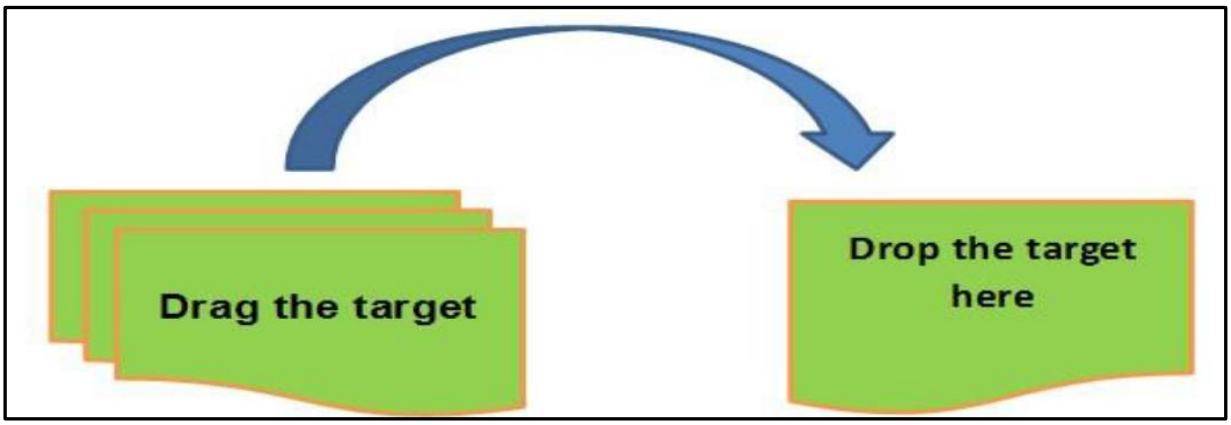


4. dragAndDrop(): This action is performed using a mouse when a user moves (drags) a web element from one location and then places (drops) it at another point. This is a common action used in Windows Explorer when moving a file from one folder to another. This is a common action used in Windows Explorer when moving a file from one folder to another. Here, the user selects a file in the folder, drags it to the desired folder, and drops it. Select some element on the web page, drag it and then place it on the alternate area.

Ex: This is a very common action used in Windows Explorer while moving any file from one folder to another.

Here, the user selects any file in the folder, drags it to the desired folder and just drops it.

To drag and drop the element first we used DragAndDrop method from the Actions class in which we pass the 2 parameters, 1st parameter is the element which we need to drag, and 2nd parameter is the element on which we need to drop the 1st element.



Syntax:

Actions action = new Actions(driver);

action.dragAndDrop(Sourcelocator, Destinationlocator).build().perform();

In the Selenium dragAndDrop method, we pass two parameters:

- The first parameter is the Sourcelocator element which is being dragged
- The second parameter is the Destinationlocator on which the previous element needs to be droppedMethods for performing drag and drop on a web element are as follows:

- **clickAndHold(WebElement element)** – Clicks a web element at the middle (without releasing)
- **moveToElement(WebElement element**) – Moves the mouse pointer to the middle of the web element without clicking
- **release(WebElement element)** – Releases the left click (which is in the pressed state)
- **build()** – Generates a composite action
- **performs**– Performs the actions

To perform the drag-drop action through a Selenium script, there is no direct drag-drop method available in WebElement interface. Unlike other commands like click(), sendKeys() there is nothing available for drag and drop. Here, we leverage the Actions class which provides various methods of emulating such complex interactions.

Methods Actions class provides for Drag-Drop action:

dragAndDrop(WebElement source, WebElement target)
dragAndDropBy(WebElementsource, int xOffset, int yOffset)

This method performs left click, hold the click to hold the source element, moves to the location of the target element and then releases the mouse click.

First, instantiate an Actions class:

Actions actions = new Actions(driver);

dragAndDrop(WebElement source, WebElement target) method has two arguments to pass. One is a source web element and another is target web element. This source web element is any web element that needs to be dragged. Target web element is any web element on which dragged object needs to be placed or dropped. To find the source and target element use the below command:

WebElement source = driver. findElement (Any By strategy & locator);

Here, you can use any By strategy to locate the WebElement like find element by its id, name attribute, etc. **actions.dragAndDrop(source,target).perform();**

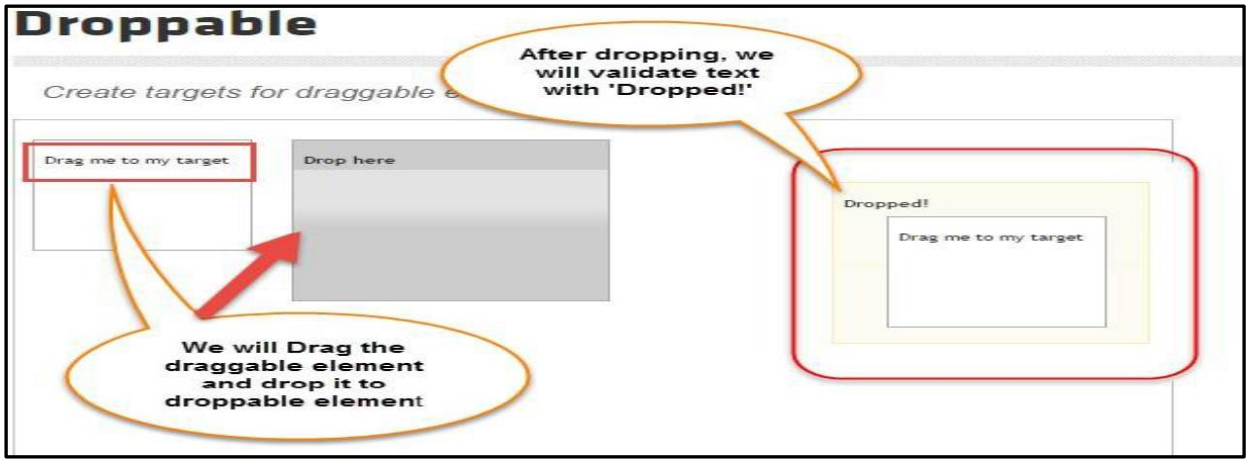
dragAndDropBy(WebElement source, int xOffset, int yOffset): method clicks & holds the source element and moves by a given offset, then releases the mouse. Offsets are defined by x & y.

As seen above, the dragAndDropBy() method has three arguments to pass. Source web element, xOffset, and yOffset.

For **Example**, if a xOffset value is set as 50, it means an object needs to be dragged and dropped by 50 pixels offset horizontal direction. Similarly, if a yOffset value is set as 50, it means an object needs to be dragged and dropped by 50 pixels offset vertical direction.

actions.dragAndDropBy(source, xOffset, yOffset).perform();

actions.dragAndDropBy(source, 50, 50).perform();



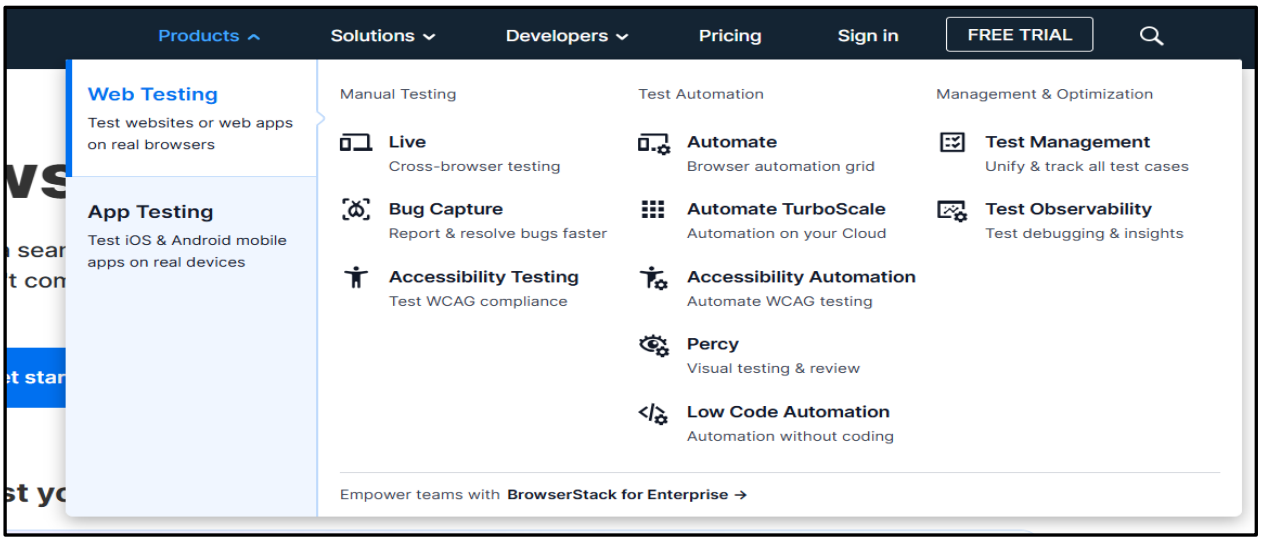
5. moveToElement: In Selenium, the moveToElement() method is a part of the Actions class. It is used to move the mouse pointer to the center of a specific WebElement. This is helpful for triggering hover effects, dropdowns, tooltips, or interacting with elements that appear when hovered over. It is a kind of action on the web page in which the mouse cursor moves to a WebElement.

To move the mouse cursor to a WebElement, we use the moveToElement() method of actions class in Selenium WebDriver.

moveToElement(WebElement toElement)

This method takes target WebElement as an input parameter on which the mouse should be moved.

```
Actions actions = new Actions(driver);
actions.moveToElement(targetElement).perform();
```



6. moveByOffset(): The moveByOffset(int xOffset, int yOffset) method in Selenium WebDriver, provided by the Actions class, is used to move the mouse pointer by a specified offset relative to its current position. This method is useful for interacting with elements based on their coordinates, such as dragging, dropping, or hovering over specific points.

The moveByOffset() method is used to move the mouse from its current position to another point on the web page. Developers can specify the x distance and the y distance the mouse has to be moved. When the page is loaded, generally the initial position of the mouse would be (0, 0), unless there is an explicit focus declared by the page.

moveByOffset(int xOffSet, int yOffSet)

xOffset: Horizontal offset in pixels (positive for right, negative for left).

yOffset: Vertical offset in pixels (positive for down, negative for up).

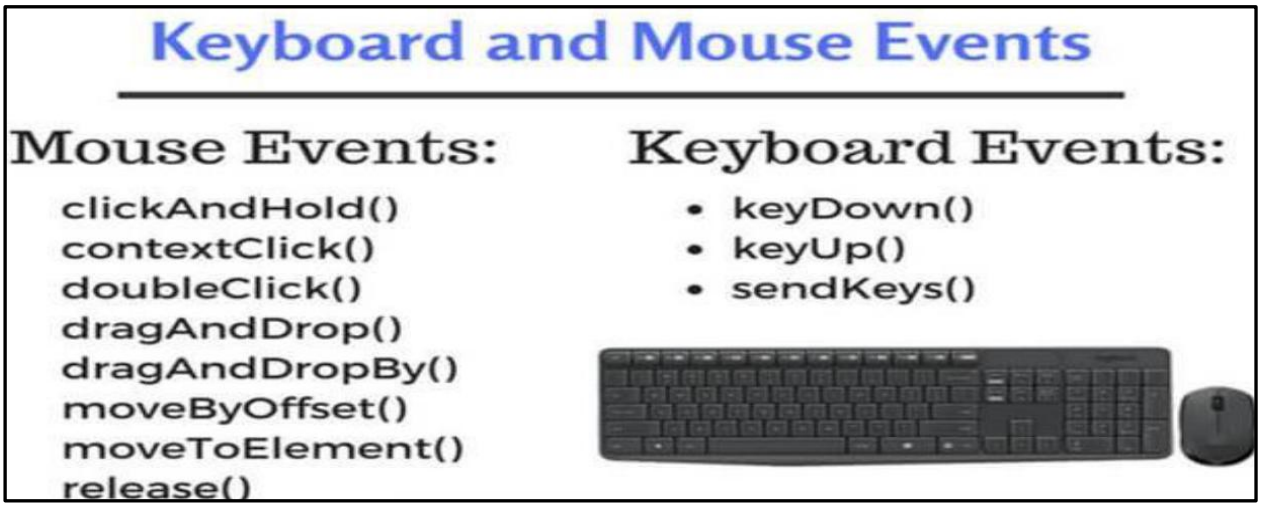
In the preceding code, xOffSet is the input parameter providing the WebDriver the amount of offset to be moved along the x axis. A positive value is used to move the cursor to the right, and a negative value is used to move the cursor to the left.

7. release(): In Selenium, the Action class provides a way to define a series of complex user interactions that simulate a user's interactions with a web page. The release() method in the Action class is used to release any system resources (e.g. keyboard, mouse) that were acquired during the performance of an action, such as clicking a button or typing into a text box.

The release() method is typically called after the action has been performed, to ensure that the system resources are freed up and can be used by other parts of the application. This

helps to avoid resource leaks, which can cause stability and performance problems over time.

In summary, the release() method in the Selenium Action class is used to free up system resources that were acquired during the performance of an action so that they can be used by other parts of the application.



****

AutoIT in Selenium

Need of third-party tools in Selenium:

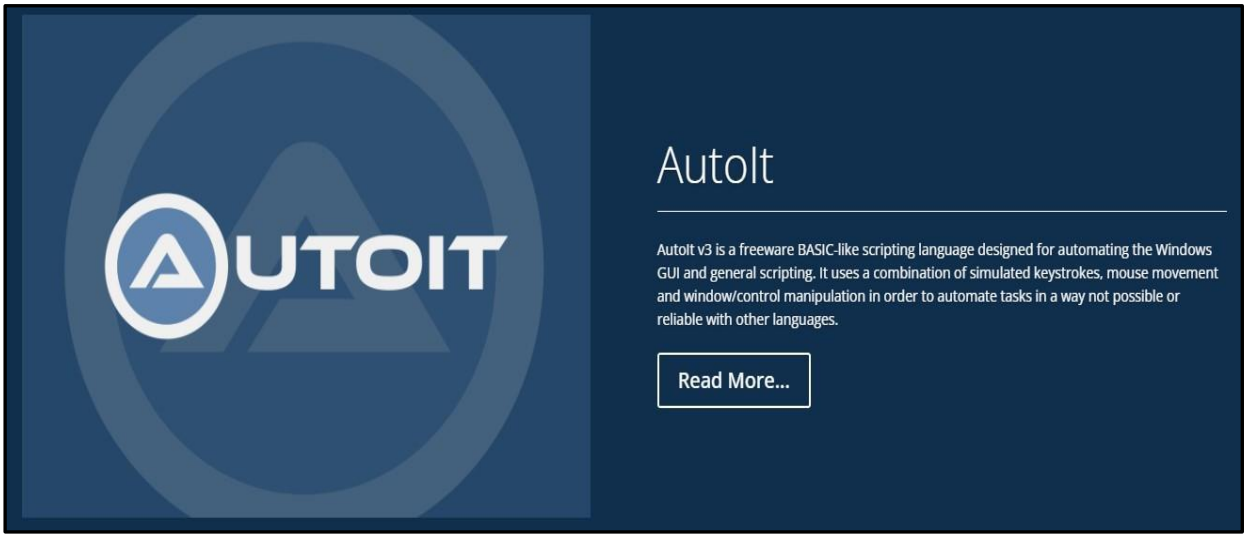
Web applications do not always confine themselves to working entirely on the web. Sometimes they need to interact with the desktop to do things like downloads & uploads. Automating these sorts of workflow is tricky in Selenium. Selenium is confined to automating browsers, so desktop windows are out of scope. If you are looking to automate workflows that go from browser to desktop and back in Selenium, then a little AutoIt is in order.

AutoIT in Selenium:

AutoIt is a popular web testing framework. AutoIt is an open-source automation language for Windows operating system. AutoIt v3 is a BASIC-like scripting computer language which is designed for automating the Microsoft Windows GUI [User Interface] and general scripting. It uses a combination of simulated keystrokes, mouse movement and window/control manipulation in order to automate tasks in a way not possible or reliable with other languages (e.g. VBScript and SendKeys). AutoIt is also very small, self-contained and will run on all versions of Windows out-of-the-box with no annoying "runtimes" required! AutoIt provides features to handle Windows dialogs, automate desktop applications, and interact with non-browser elements.

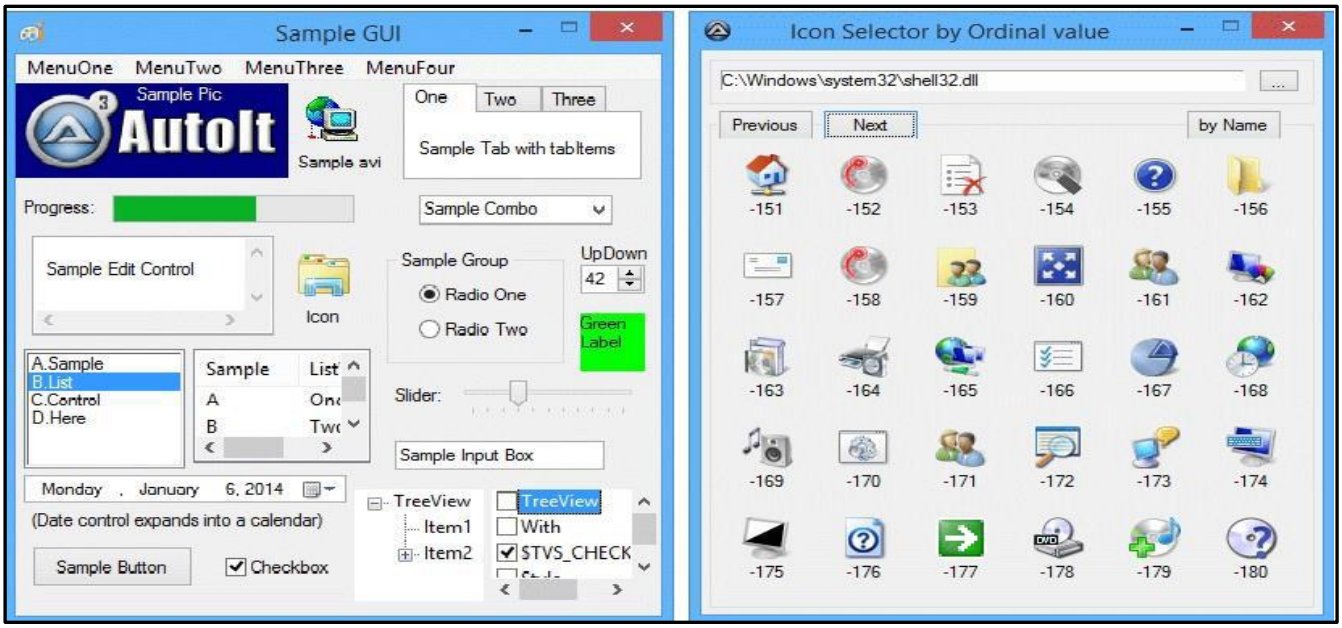
In general term AutoIt is just another automation tool like Selenium but unlike Selenium it is used for Desktop Automation rather Web Automation. It is a powerful tool and it just not automate desktop windows, button & form, it automates mouse movements & keystrokes too. Just like Selenium IDE, it also gives you the recording capability which generates the scripts for you to use the same script in you test.

The scripts which are written using AutoIt can be converted into a compressed and stand-alone executable(.exe) which can be run on computers that do not have the AutoIt interpreter installed. AutoIt also has IDE based on SciTE editor. The compiler and help text are integrated and it provides a standard environment for developers.



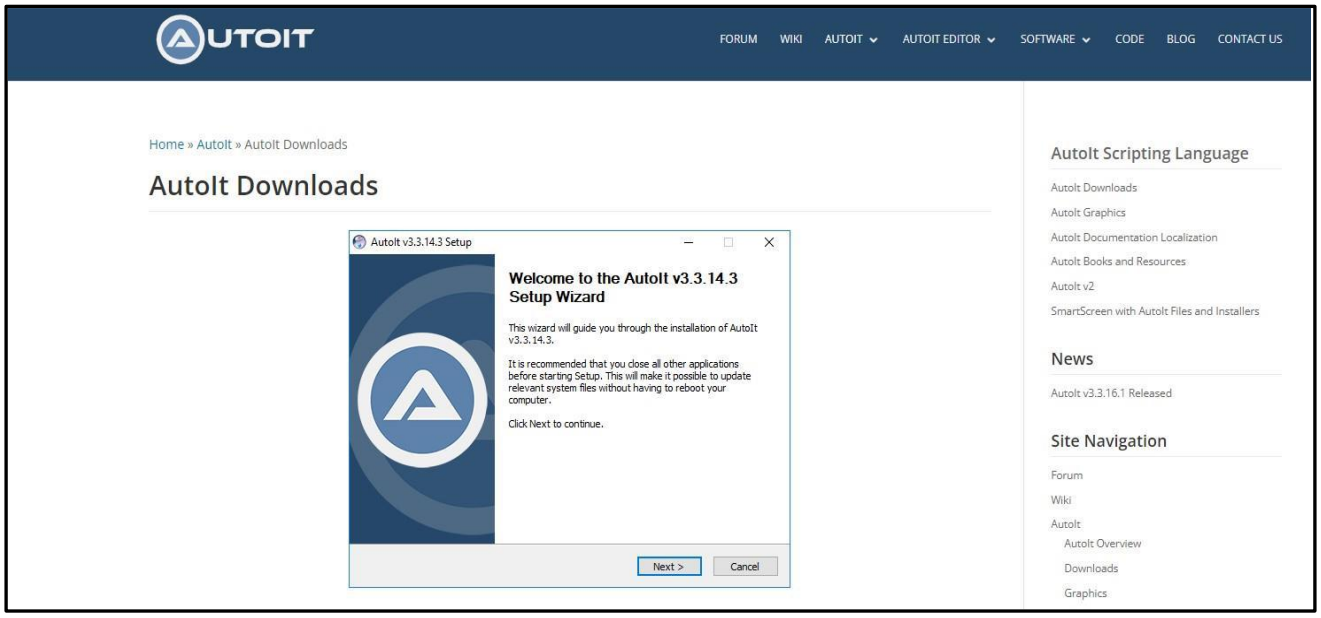
AutoIt has the following Features:

1. **BASIC-like Syntax and Rich Function Set:** It has a BASIC-like syntax. This means people who have knowledge of scripting or used a high-level language should be able to understand it easily.
2. **AutoIt contains a built-in Editor with Syntax Highlighting:** It means AutoIt comes with a customised "lite" version of SciTe that makes editing scripts easy.
3. **Desktop Automation:** AutoIt provides a way to automate tasks in desktop applications, such as filling out forms, clicking buttons, and navigating through menus. This allows testers to automate complex scenarios that involve interacting with desktop applications, providing greater coverage for their automated tests.
4. **Standalone and Small:** AutoIt is a small and standalone application. Download AutoIt executable (AutoIt3.exe) to run AutoIt scripts.
5. **Key and Mouse Simulation:** This is the main feature of AutoIt as it has a good keystroke and mouse simulation functions. All the keyboard and mouse routines are highly configurable as they work in "speed" and functionality.
6. **Image Recognitio**n: AutoIt has built-in image recognition capabilities, which allow testers to interact with elements in desktop applications based on their visual appearance.
7. **Controls:** AutoIt directly gets information on and also interacts with edit boxes, list boxes, check boxes, buttons, combos, and status bars without the risk of keystrokes getting lost.
8. **Autoit in Selenium:** AutoIt can be seamlessly integrated with Selenium scripts, allowing testers to combine the strengths of both tools to automate a wide range of tasks in web applications. This integration allows testers to leverage the features of AutoIt, such as handling Windows dialogs and automating desktop applications, in conjunction with Selenium's web testing capabilities.
9. **Graphical User Interfaces (GUIs):** AutoIt v3 will also allow you to create some complex GUIs - just like those below!

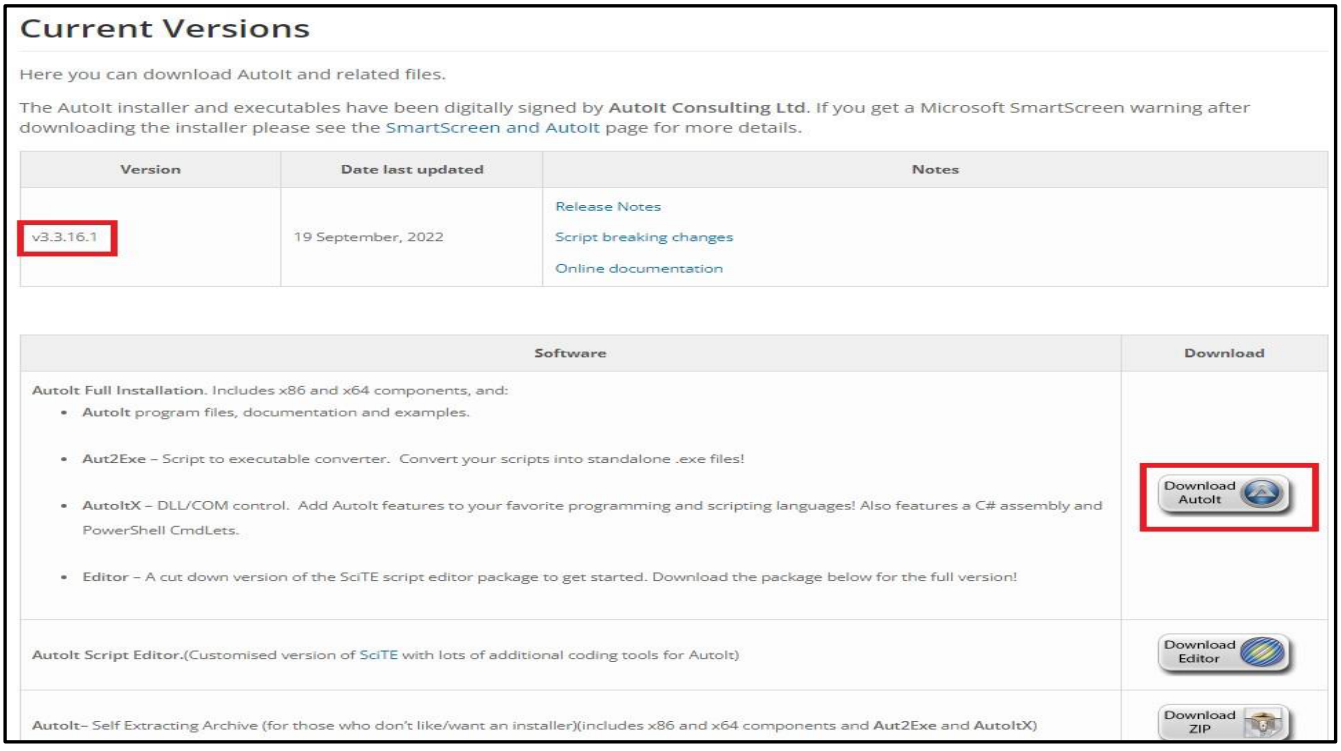


How to download and install AutoIT:

Step 1: Go to AutoIt website and navigate to download page. It will display the latest version. The top most download is for AutoIt, click on it. Go to this link to download AutoIt website [https://www.autoitscript.com/site/autoit/downloads].

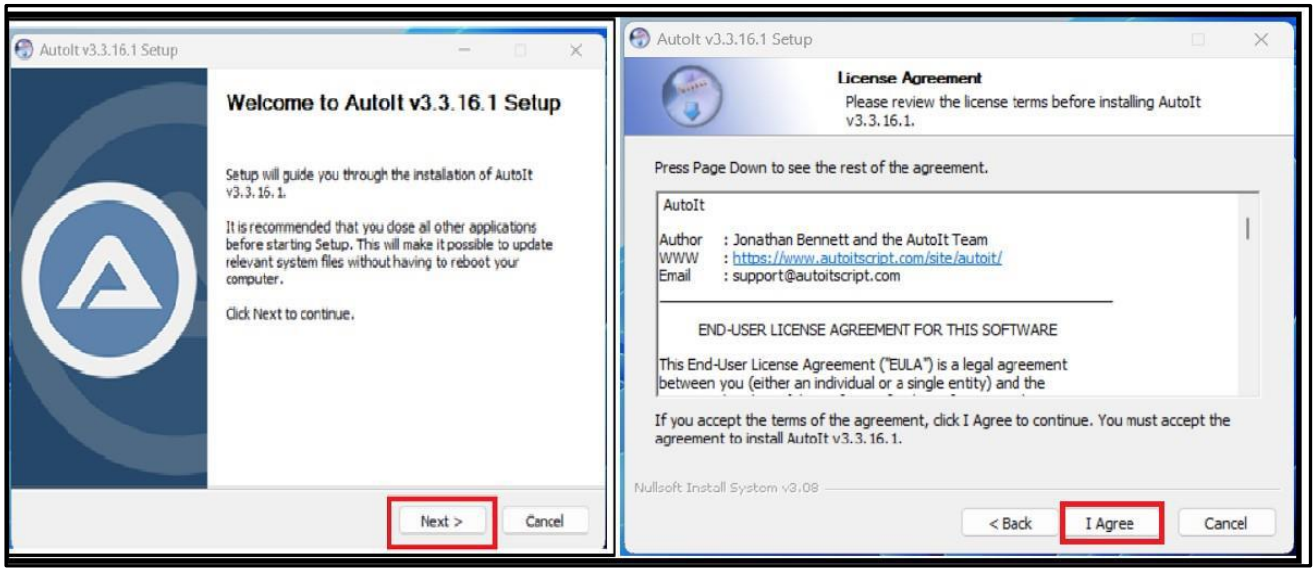


Step 2: Download "Autoit" by clicking on 'Download Autoit' button [Scroll down and Click on the "Download AutoIt" button].

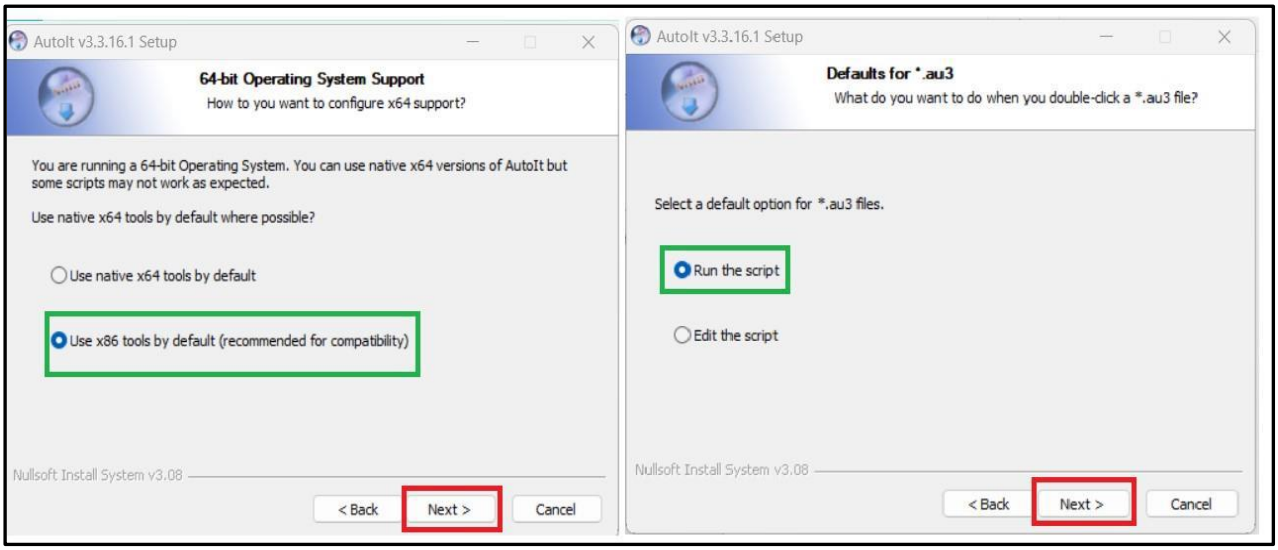


Step 3: Once the download is complete, double-click on the downloaded executable file to start the installation process.

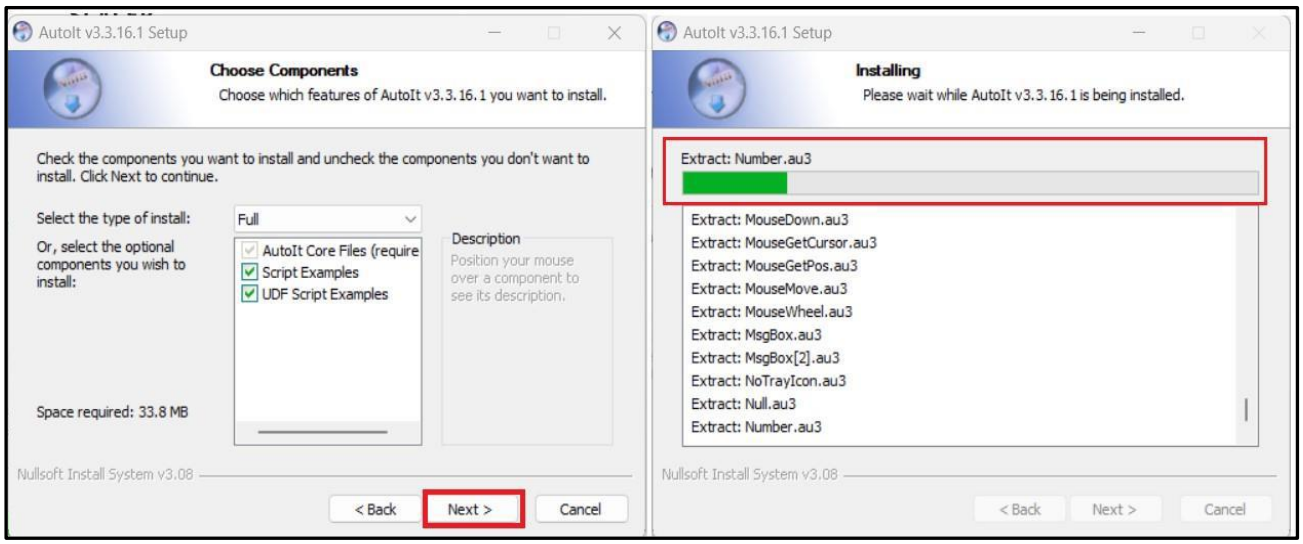
Step 4: Click on Next, accept all the default settings, and hit on the I agree button. Click on Install.



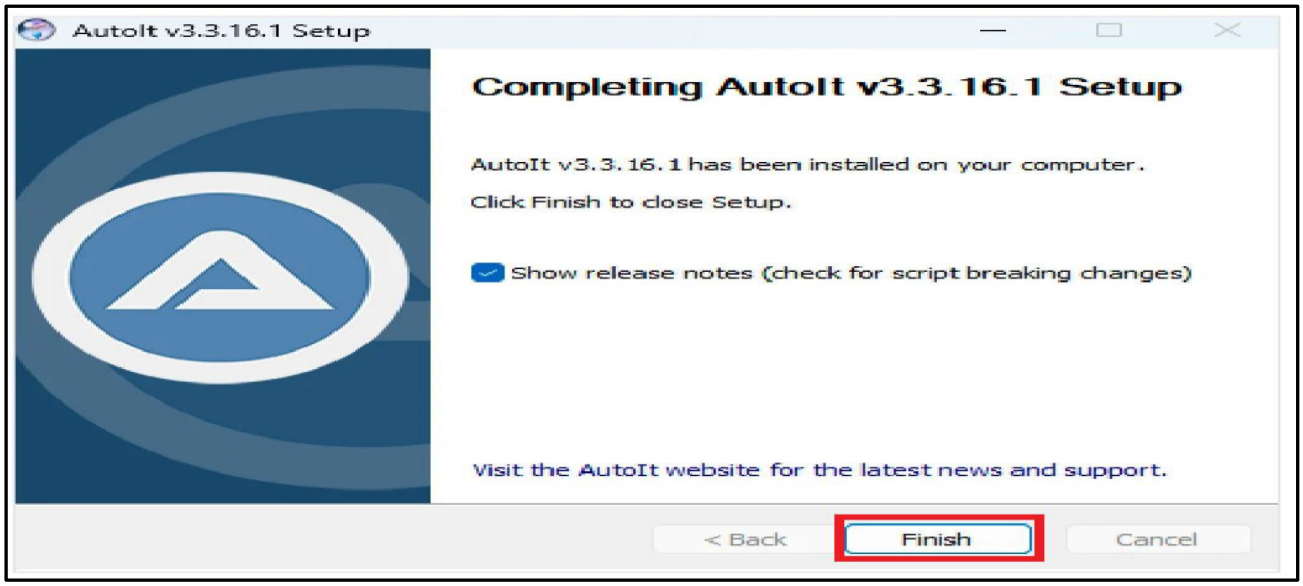
Step 4: Follow the instructions in the setup wizard to install AutoIt on your system. The installation starts.



Step 5: Follow the instructions in the setup wizard to install AutoIt on your system. The installation starts.

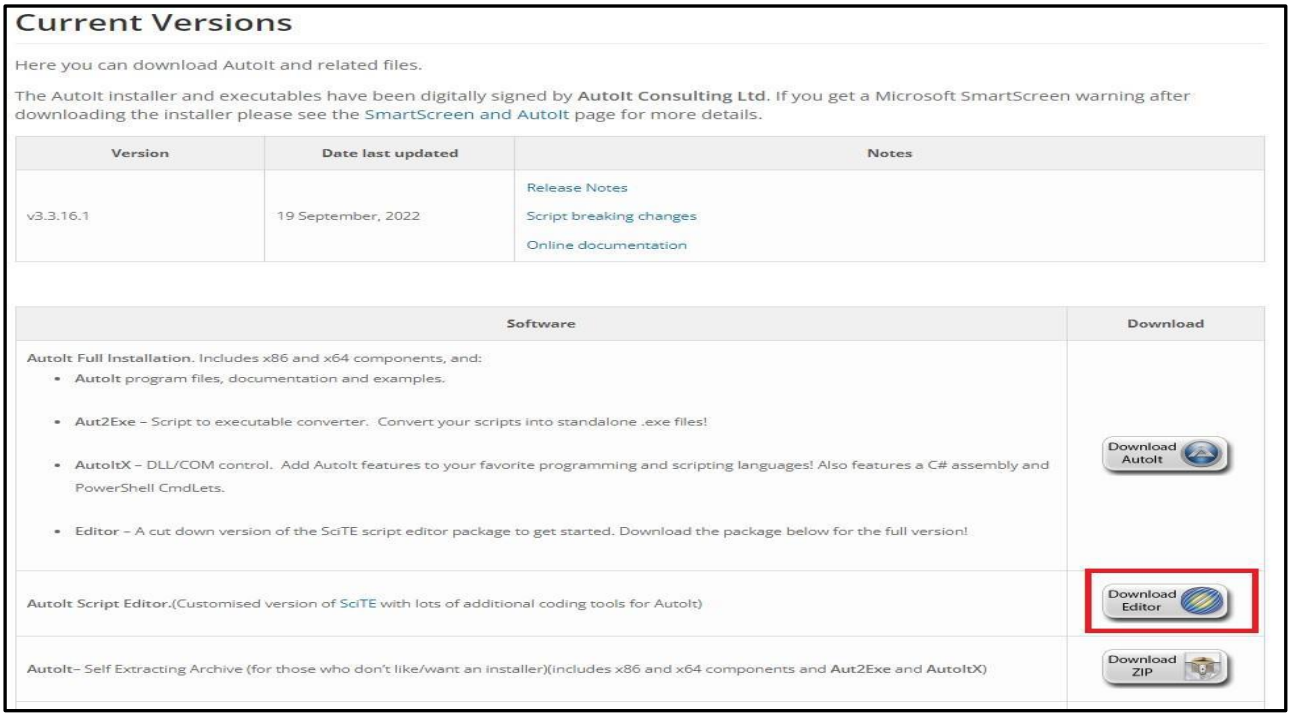


Step 6: Click on Finish. AutoIt is installed in the system.

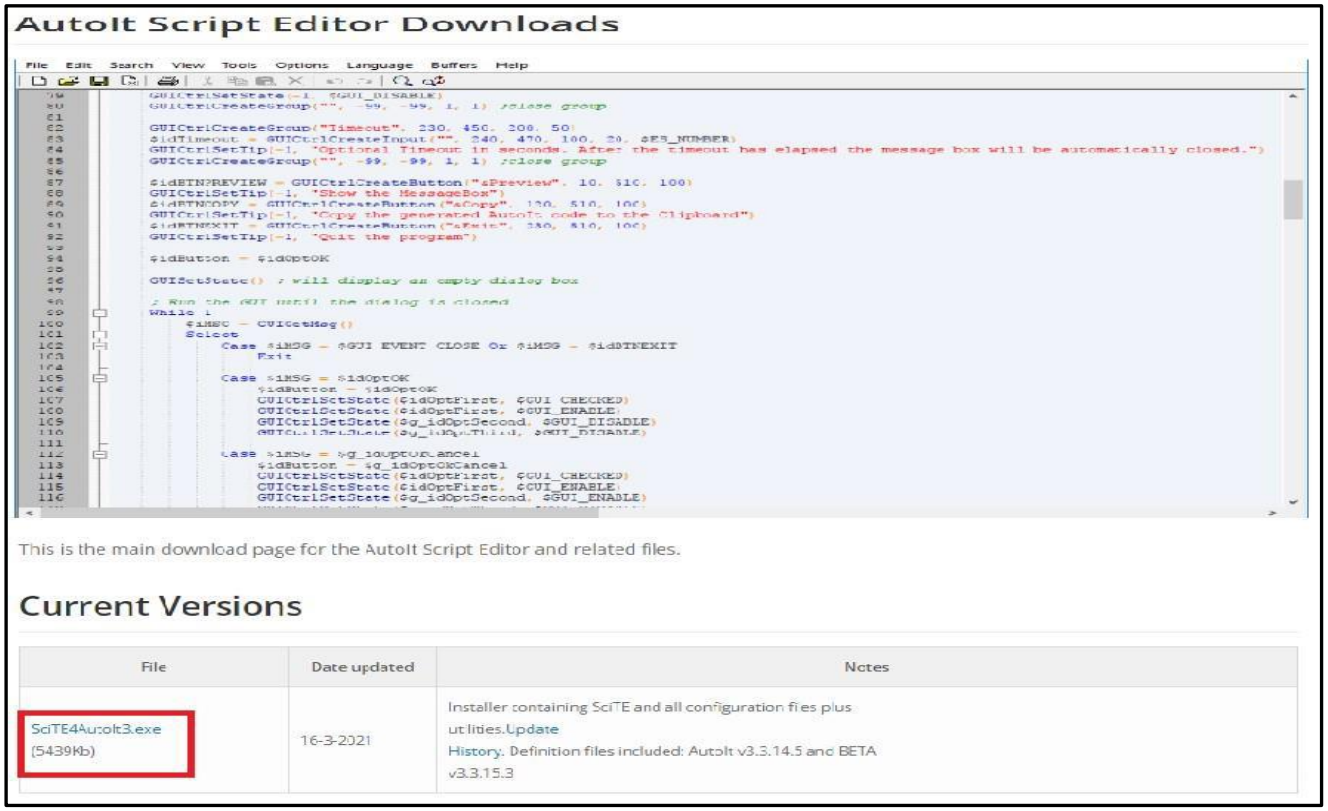


How to download and install AutoIt Script Editor:

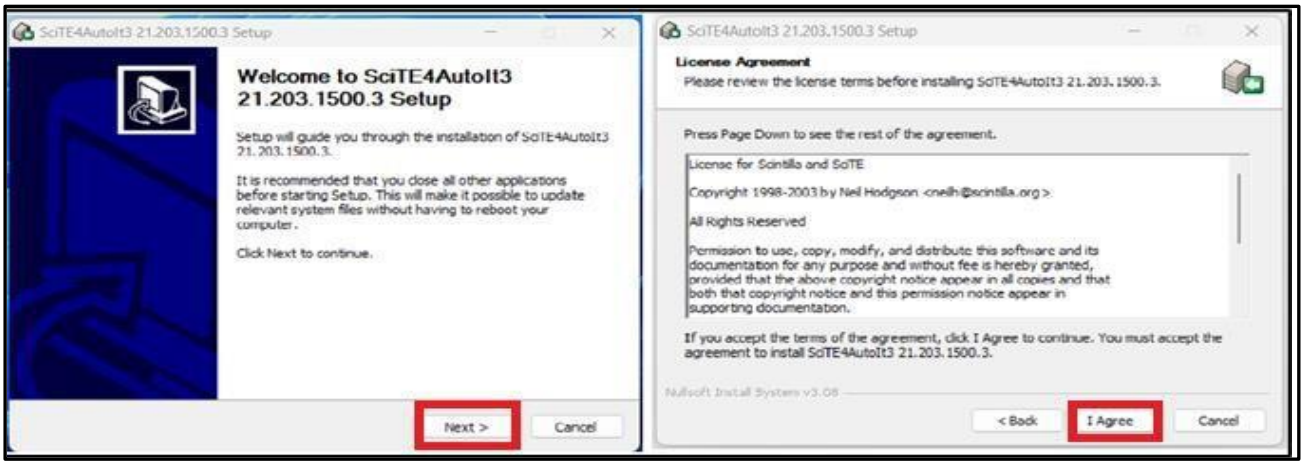
Step 1: Now, we have to download the AutoIt script editor. Click on this link [https://www.autoitscript.com/site/autoit-script-editor/downloads] to open the AutoIt editor website.



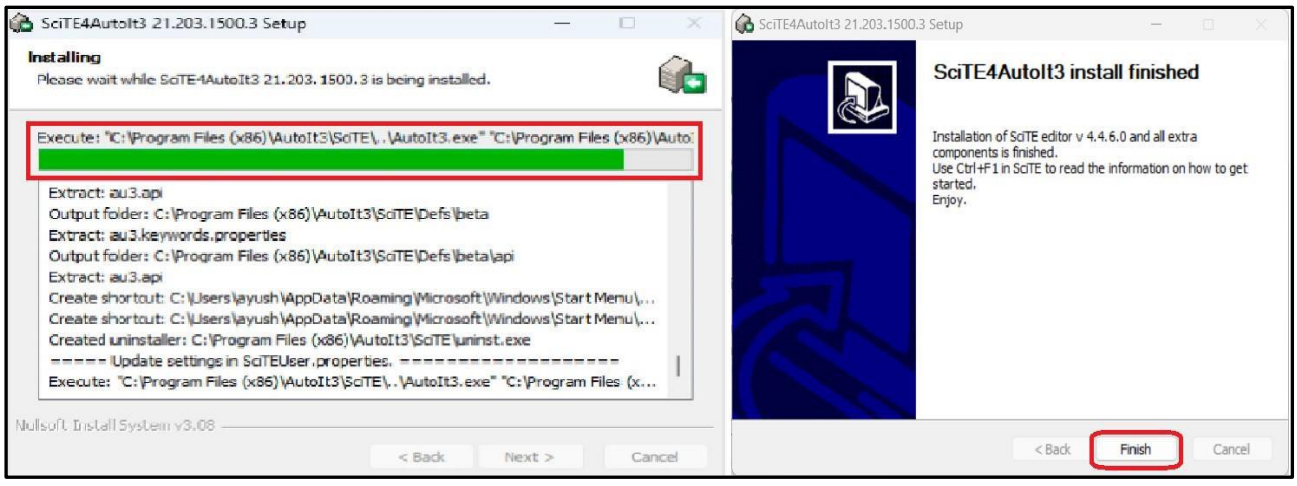
Step 2: Scroll down and Click on the SciTE4AutoIt3.exe to download the editor.



Step 3: Open the file to install the editor. Click on Next and agree to the conditions. Then click on Install.

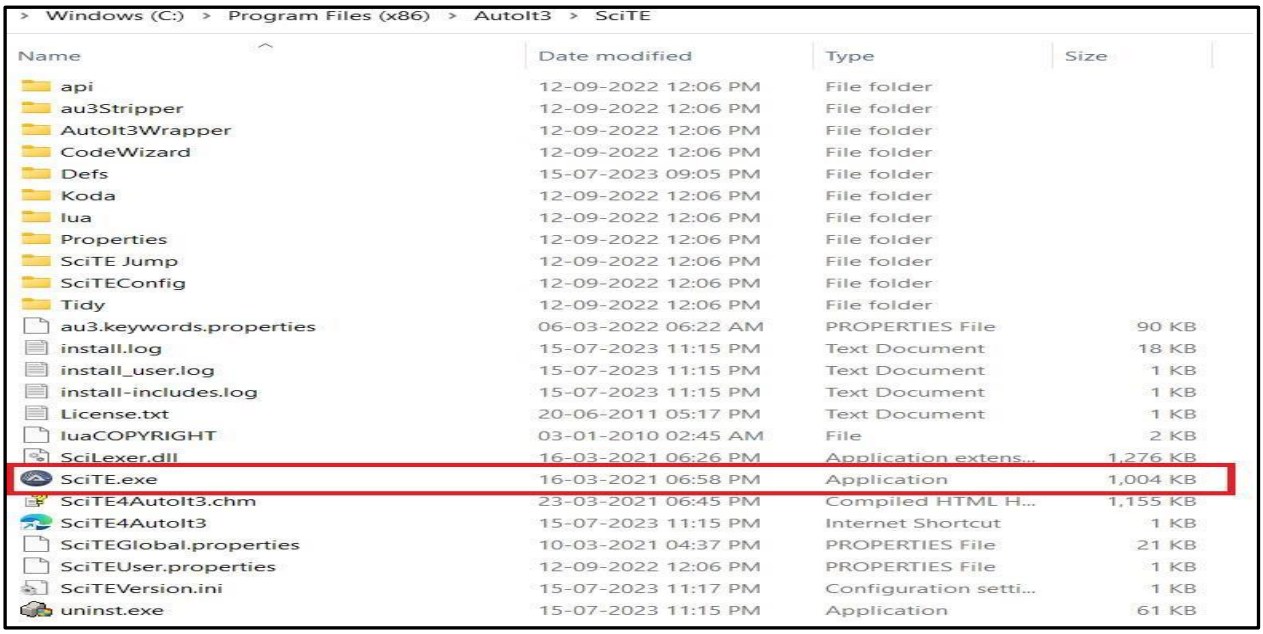


Step 4: It will start installing the setup of the autoit editor. Click on Finish once done.

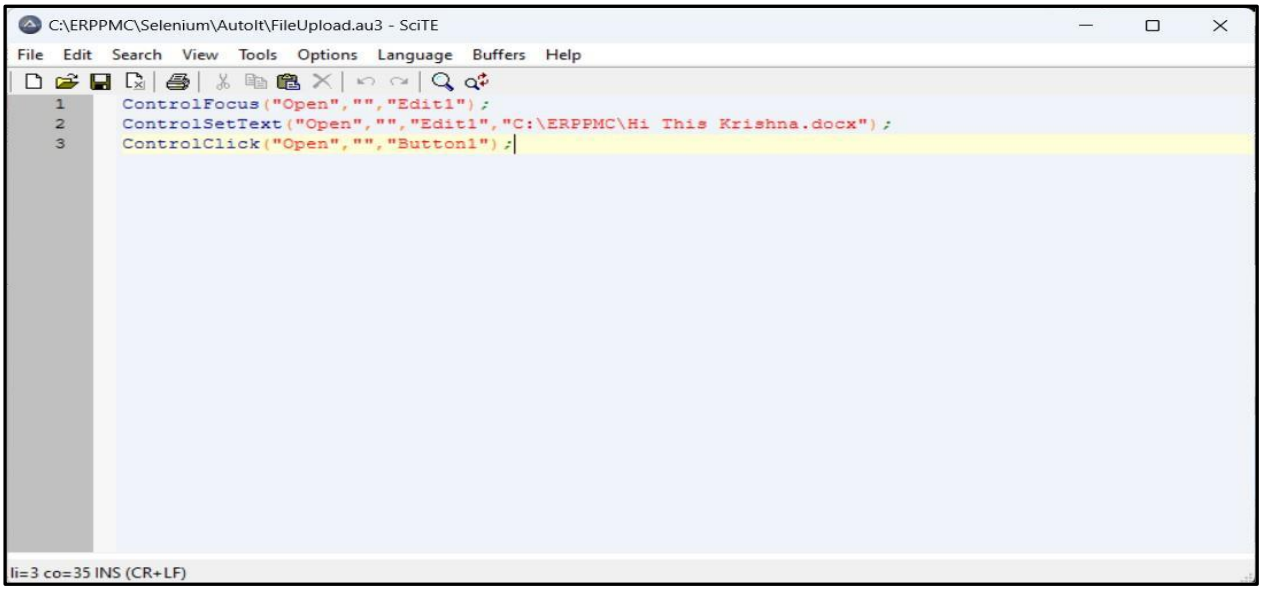


After successfully installation – open up AutoIT Editor.

[C:\Program Files (x86)\AutoIt3\SciTE]

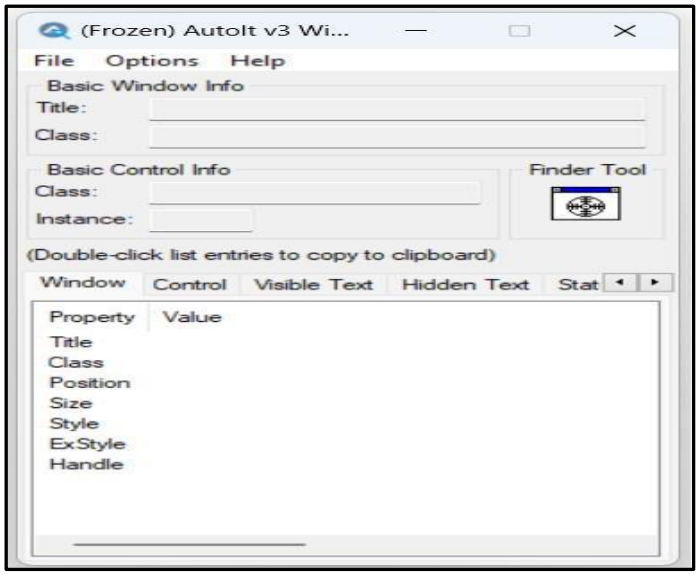


Click on 'SciTE.exe' file, the AutoIT editor opens as shown in below screen.



Now opens element Identifier [C:\Program Files (x86)\AutoIt3].

Click on ‘Au3Info.exe’ file, the element identifier opens as shown in below screen.

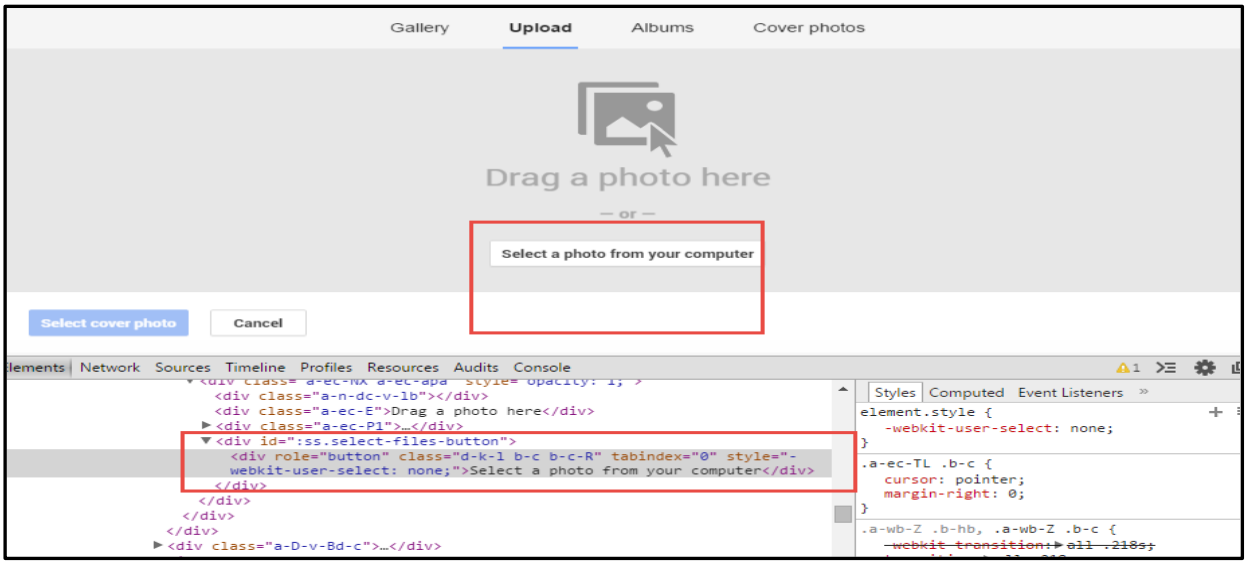


How to Use AutoIT with Selenium Webdriver:

Upload file in Selenium using AutoIt:

If there is no text box to set the file path and only able to click on Browse button to upload the file in the windows popup box then we do upload file using AutoIt tool. The following procedures must be followed in order to use AutoIT with Selenium WebDriver for a file upload. The AutoIT script you create will handle the file upload dialogue box. After setting the file location and clicking the "Open" button, the script must wait for the dialogue to appear.

The below is the sample HTML source code with upload button:



If you observe the above screen shot, there are is no 'Input' html tag. Button is completely customized using css. Here we cannot send the file path using sendKeys method. Hence, we should go for other Tool support to handle Operating System events.

Uploading of file is a four steps process:

Step 1: Identify the Windows control.

Step 2: Build a AutoIt script using identified windows control.

Step 3: Compile the .au3 script and convert it in to .exe file.

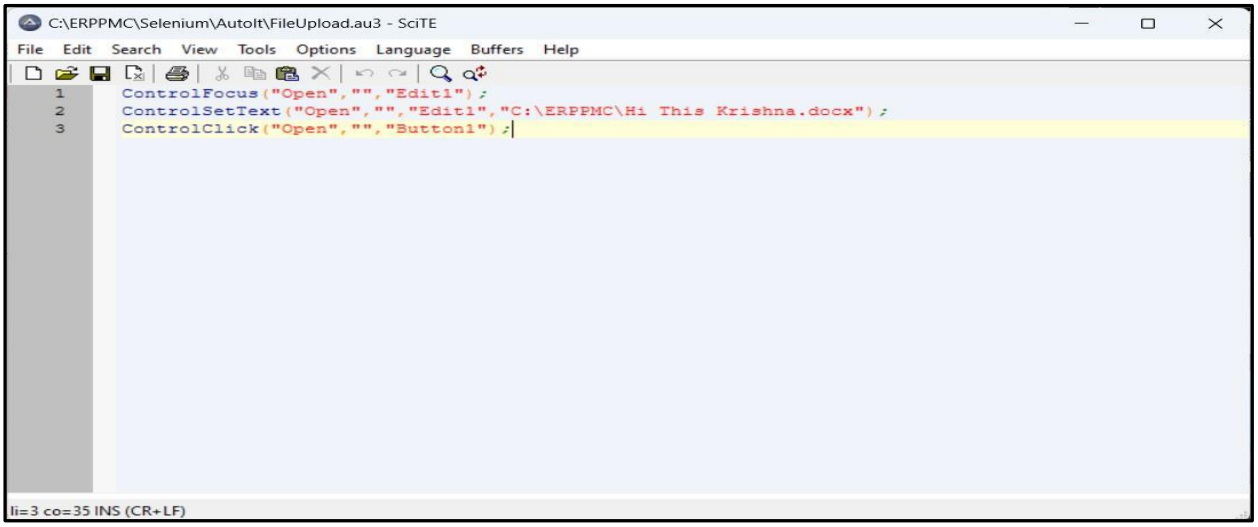
Step 4: Call the .exe file in to the Selenium test case.

Step 1: Identify the Windows control:

Navigate to Practice Form page of ToolsQA [https://demoqa.com/automation-practiceform]

Click on 'Browse' button of Profile Picture column. It will open a Windows box for 'File Upload'.

Now go to your [C:\Program Files (x86)\AutoIt3\SciTE] AutoIt v3 and open 'SciTE Script Editor'. It will open an editor window where we write out automation script of AutoIt. The AutoIT editor opens as shown in the below screen.



AutoIT Script for file Uploading (AutoIT Editor) in Selenium WebDriver:

AutoItScript Explanation:

ControlFocus(" title "," text ",controlID)

Above the line of code changes the focus to the file upload windows.

ControlSetText(" title "," text ",controlID ," File path which need to upload ") set text/path into file name edit box.

ControlClick(" title "," text ",controlID) click open to upload file.

Step 1: Open SciTE Script editor and add the below mentioned AutoIt script and save it as 'UploadFile.au3' in your system. **AutoItScript:**

ControlFocus("Open","","Edit1") [ControlFocus() --> focus on the text box]

ControlSetText("Open","","Edit1","C:\Users\PrajjawalK\Music\check\visit.pdf") [ControlSetText() --> providing path of a file]

ControlClick("Open","","Button1") [ControlClick() --> clicking on open button]

 After writing above AutoIt script in Autoit editor and save it as "**UploadFile.au3**" [Filename. Extension] in your system. Once the file is saved.

Compile AutoIT script and generate .exe file

Tools-->Compile-->Select x64--> Compile --> generated .exe file **Use and integrate .exe file in selenium webdriver script:**

Ex: Runtime.getRuntime().exec("C://autoitfiles/fileupload.exe);

****

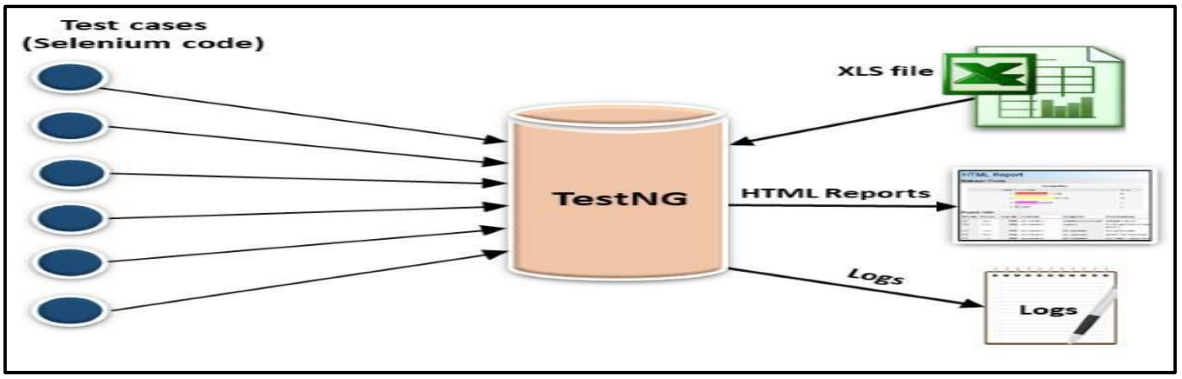
TestNG Frame Work

TestNG is an automation testing framework in which NG stands for "Next Generation". TestNG is a testing framework inspired from Junit [Java unit testing framework] and NUnit [unit testing framework for the .NET] but introducing some new functionalities that make it more powerful and easier to use. It is considered as an upgraded version of these two frameworks as it provides additional functionalities like test annotations, prioritizing tests in TestNG, grouping, parameterization and sequencing techniques in the code – which were not possible earlier. TestNG is inspired by JUnit which uses the annotations [@]. TestNG overcomes the disadvantages of JUnit and is designed to make end-to-end testing easy. Default Selenium tests do not generate a proper format for the test results. Using TestNG in Selenium, we can generate test results. Using TestNG, you can generate a proper report, and you can easily come to know how many test cases are passed, failed, and skipped. You can execute the failed test cases separately.

- ✓ It is an open-source automated TestNG framework. In TestNG, NG stands for "Next Generation".
- ✓ It permits many threads to run at the same time and simultaneously.
- ✓ TestNG reuses similar test class instances (just like Junit) for all kinds of test methods.
- ✓ TestNG offers a lot of flexibility in the manner in which it passes parameters into different unit tests.
- ✓ Another feature that keeps this framework apart is its ability to put test methods in Java groups.
- ✓ In addition, The TestNG framework performs unit testing, end-to-end testing, integration testing, etc.

For example:

Suppose, you have five test cases, one method is written for each test case (Assume that the program is written using the main method without using testNG). When you run this program first, three methods are executed successfully, and the fourth method is failed. Then correct the errors present in the fourth method, now you want to run only fourth method because first three methods are anyway executed successfully. This is not possible without using TestNG. The TestNG in Selenium provides an option, i.e., testng-failed.xml file in test-output folder. If you want to run only failed test cases means you run this XML file. It will execute only failed test cases.



TestNG is an automation testing framework in which NG stands for "Next Generation". TestNG is inspired by JUnit which uses the annotations (@). TestNG overcomes the disadvantages of JUnit and is designed to make end-to-end testing easy.

JUNIT Testing Framework:

Junit [Java Unit] is the most popular Java unit testing framework. JUnit is an open-source unit testing framework for java programmers. It is useful for Java Developers to write and run repeatable tests. It is only used for unit testing. That leads to better and faster development from developers across the world.

Features of JUnit:

- JUnit is an open-source framework, which is used for writing and running tests.
- Provides annotations to identify test methods.
- Provides test runners for running tests.
- JUnit tests allow you to write codes faster, which increases quality.
- JUnit is very simple. It is less complex and takes less time.
- JUnit tests can be run automatically and they check their own results and provide immediate feedback. There's no need to manually comb through a report of test results.
- JUnit shows test progress in a bar that is green if the test is running smoothly, and it turns red when a test fails.

NUnit Testing Framework:

NUnit is a unit-testing framework for all .Net languages. NUnit is Open-Source software and NUnit 3.0 is released under the MIT license. This framework is very easy to work with and has user friendly attributes for working. Initially ported from JUnit, the current production release, version 3, has been completely rewritten with many new features and support for a wide range of .NET platforms. It's completely written in C# and has been totally rewritten to reap the benefits of many of the.NET language's features, such as custom attributes and other reflection-related features. NUnit adds xUNit support to all.NET languages.

Features of NUnit:

- NUnit is a free open-source testing framework for.NET projects.
- NUnit is a framework for automating unit testing in all.Net languages.
- Traditionally adapted from Junit, which is a Cross-platform unit testing framework.
- It has been extensively rebuilt to take advantage of many.NET language capabilities and is written entirely in C#.

- NUnit adds xUNit support to all tests written in a.NET supported language, such as C#, VC, VB.NET, J#, and others.
- NUnit is a user interface program that allows you to run test scripts and see the results.
- To make unit testing easy and present the results as a success or failure, we must develop test scripts and use NUnit tools and classes.

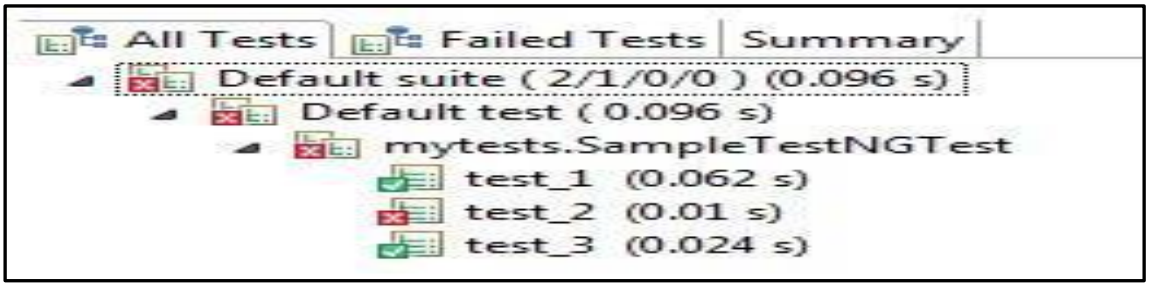
Advantages of TestNG over JUnit:

- ✓ Annotations are easier to understand.
- ✓ Parallel testing is possible.
- ✓ TestNG enables you to group the test cases easily which is not possible in JUnit.
- ✓ In TestNG, annotations are easier to understand than Junit.
- ✓ It produces the HTML reports for implementation.
- ✓ It also generates the Logs.
- ✓ TestNG enables you to group the test cases easily which is not possible in JUnit.
- ✓ TestNG supports three additional levels such as @Before/After suite, @Before/AfterTest, and Before/AfterGroup.
- ✓ It does not extend any class. Therefore, the TestNG framework allows you to define the test cases where each test case is independent of other test cases.
- ✓ Parallel execution of test cases, i.e., running multiple test cases is only possible in the TestNG framework.

Features of Selenium TestNG:

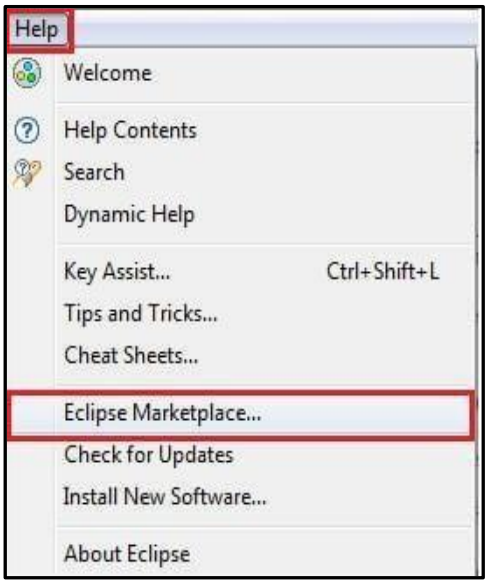
- ✍ Generate the report in a proper format including a number of test cases runs, the number of test cases passed, the number of test cases failed, and the number of test cases skipped.
- ✍ Multiple test cases can be grouped more easily by converting them into testng.xml file. In which you can make priorities which test case should be executed first.
- ✍ The same test case can be executed multiple times without loops just by using keyword called 'invocation count.'
- ✍ Using testing, you can execute multiple test cases on multiple browsers, i.e., cross browser testing.

- The TestNG framework can be easily integrated with tools like TestNG Maven, Jenkins, etc.
- Annotations used in the testing are very easy to understand ex: @BeforeMethod, @AfterMethod, @BeforeTest, @AfterTest.
- TestNG simplifies the way the tests are coded. There is no more need for a static main method in our tests.
- WebDriver has no native mechanism for generating reports. TestNG can generate the report in a readable format like the one shown below.



TestNG Installation in Eclipse:

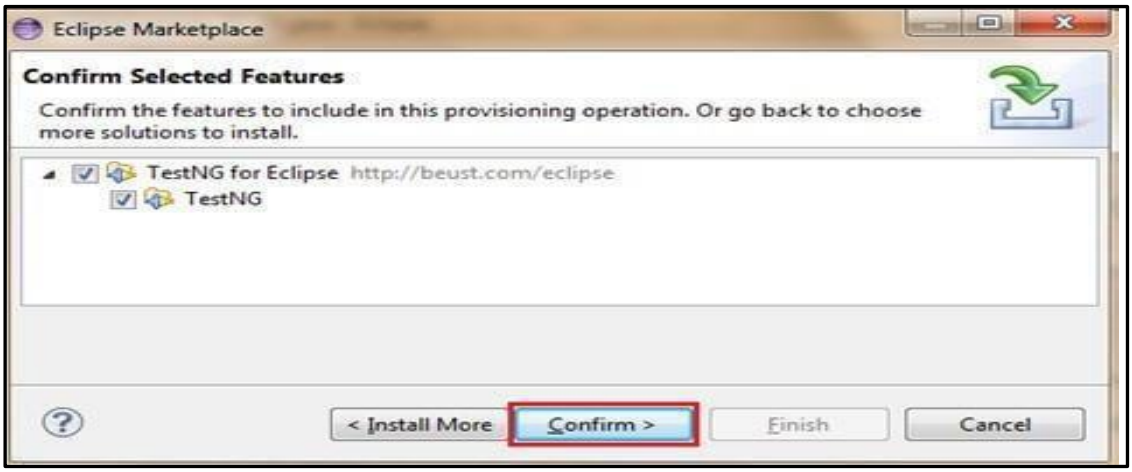
Step 1: Launch eclipse IDE -> Click on the Help option within the menu -> Select "Eclipse Marketplace" option within the dropdown.



Step 2: Enter the keyword "TestNG" in the search textbox and click on "Go" button as shown below.

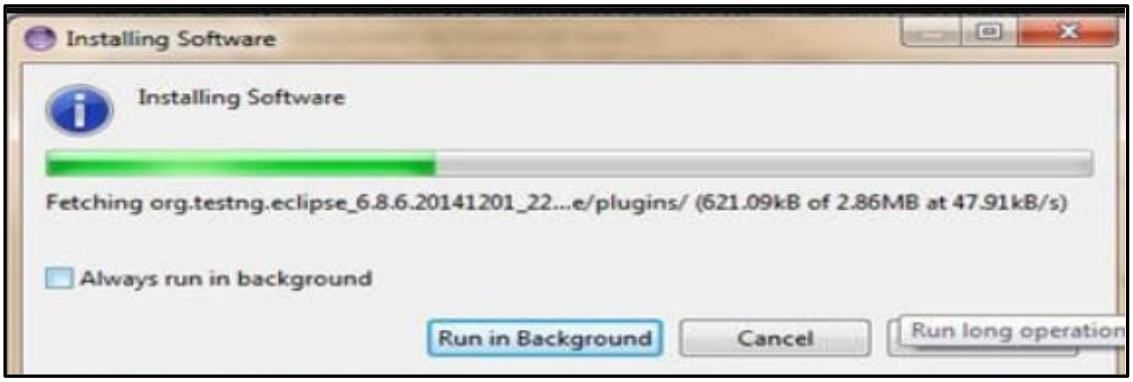
Step 3: After clicking on "Go" button it displays a "TestNG for Eclipse". Now user can click on the Install button to install TestNG.

Step 4: As soon as the user clicks on the Install button, the user is prompted with a window to confirm the installation. Click on “Confirm” button.

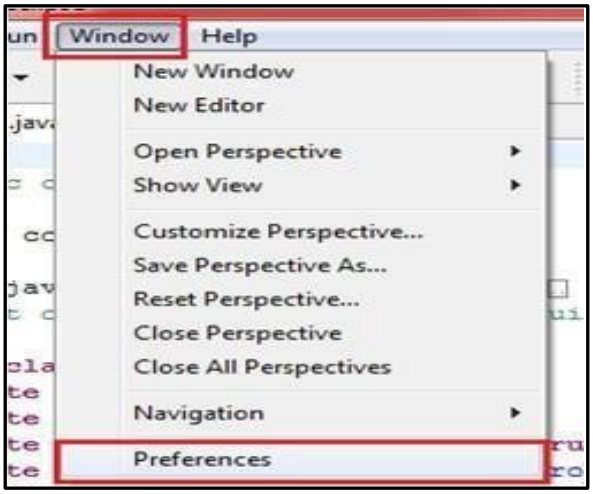


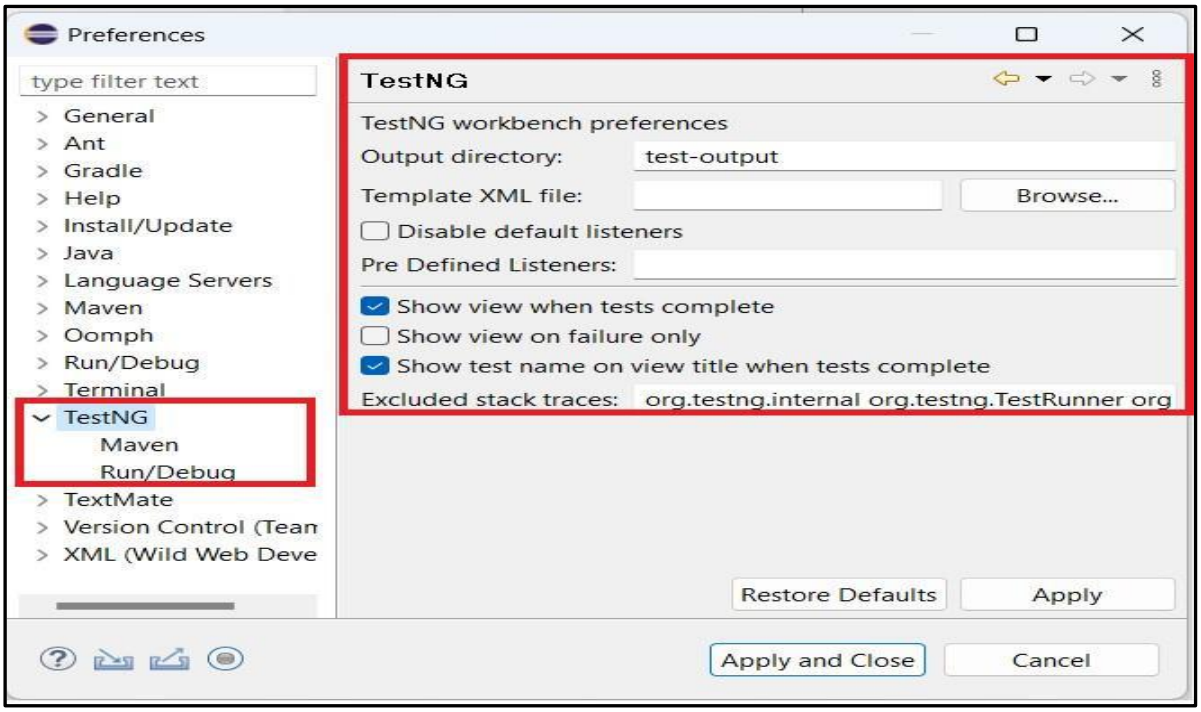
Step 5: In the next step, the application would prompt you to accept the license and then click on the “Finish” button.

Step 6: The installation is initiated now and the progress can be seen as follows:



Step 7: After successfully installation restart our eclipse so as to reflect the changes made. Upon restart, a user can verify the TestNG installation by navigating to “Preferences” from “Window” option in the menu bar. Refer the following figure for the same.

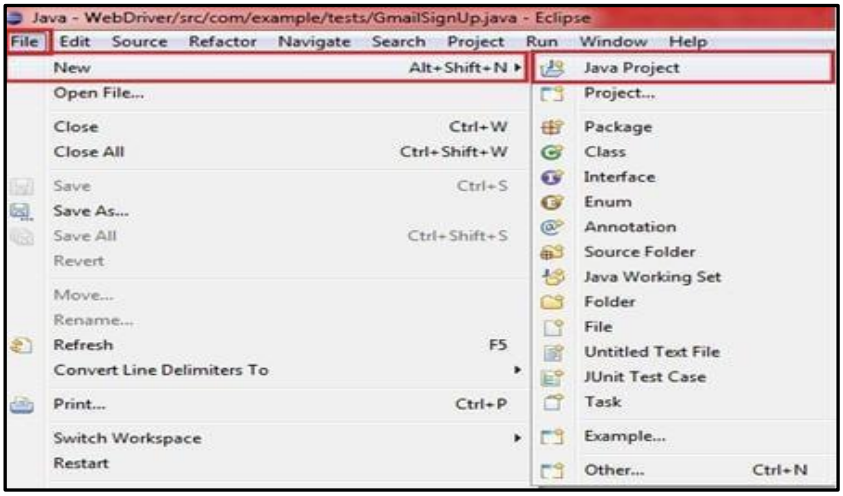




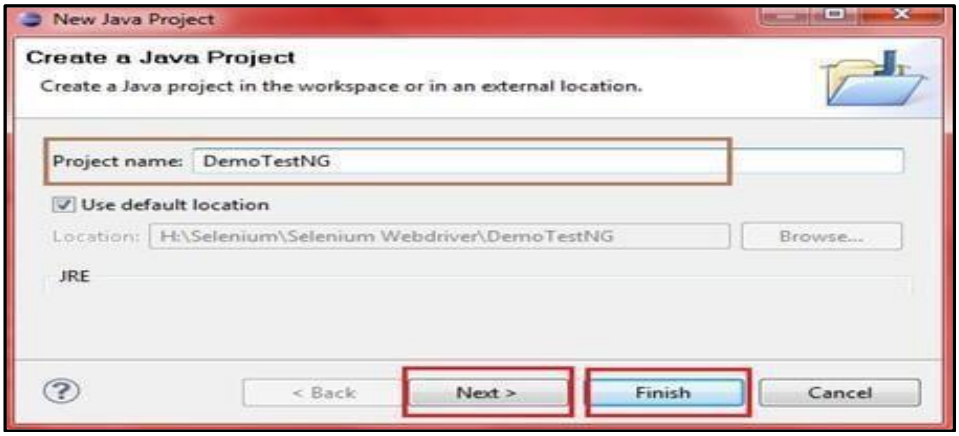
Creation Of Sample TestNG Project:

Let us begin with the creation of TestNG project in eclipse IDE.

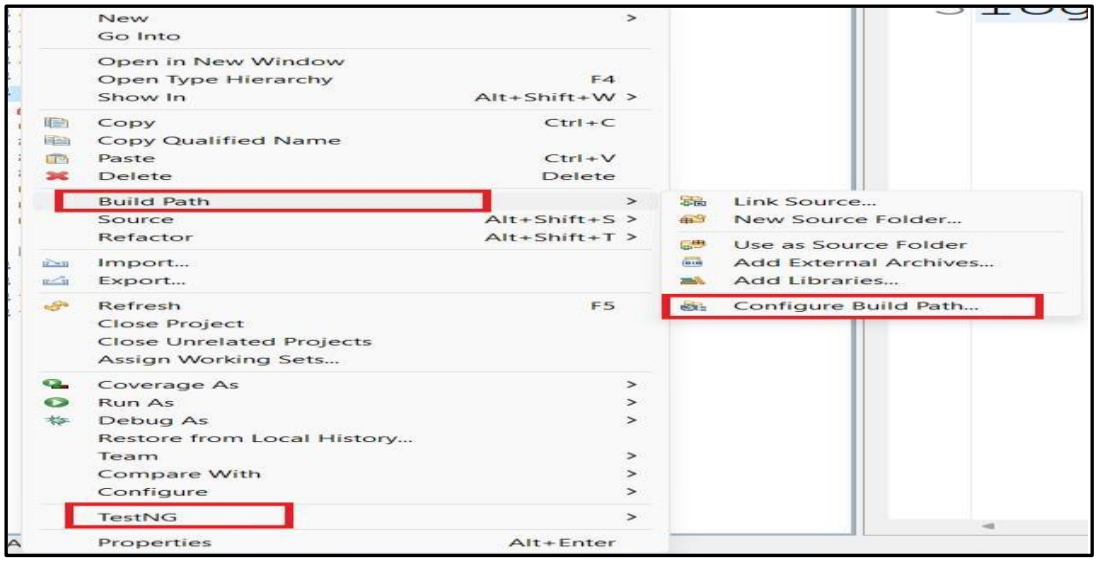
Step 1: Click on the File option within the menu -> Click on New -> Select Java Project.

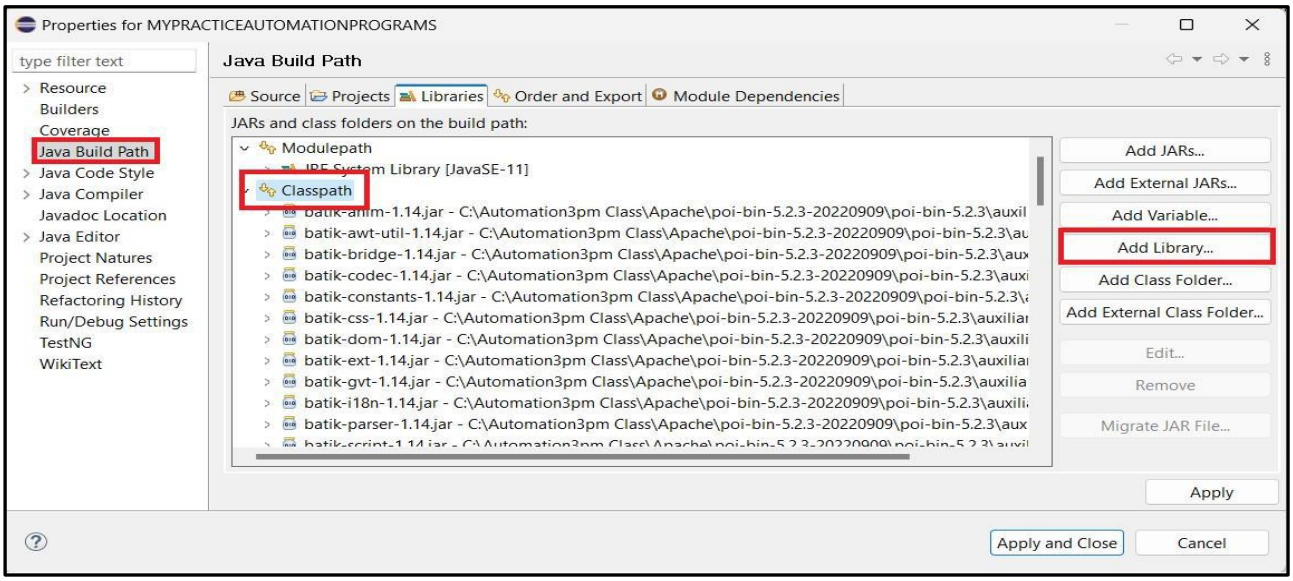


Step 2: Enter the project name as "DemoTestNG" and click on "Next" button. As a concluding step, click on the "Finish" button and your Java project is ready.

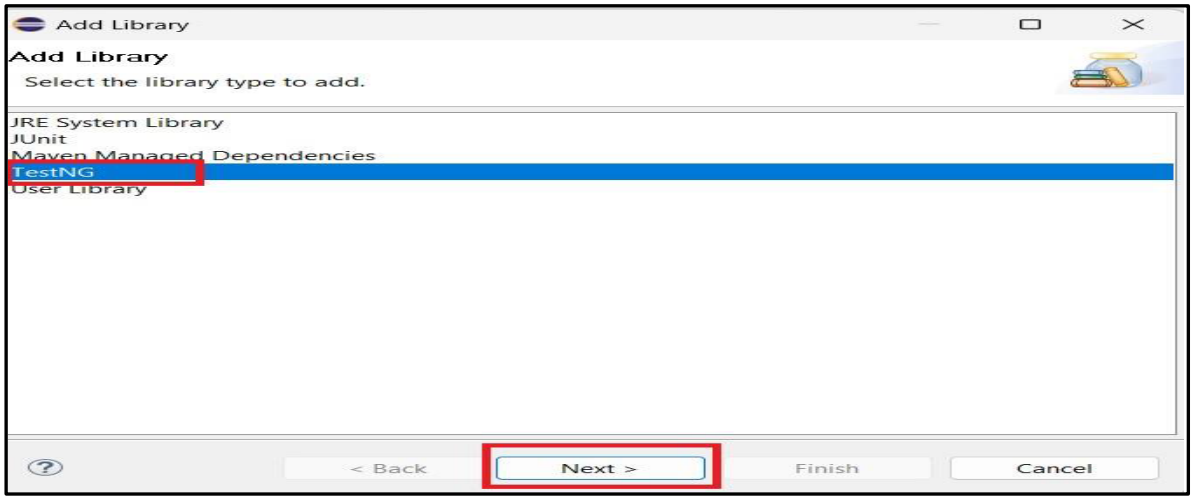


Step 3: The next step is to configure the TestNG library into the newly created Java project. For the same, click on the "Libraries" tab under Configure Build Path. Click on "Add library" as shown below.





Step 4: Select TestNG and click on the "Next" button as shown below in the image. In the end, click on the "Finish" button.

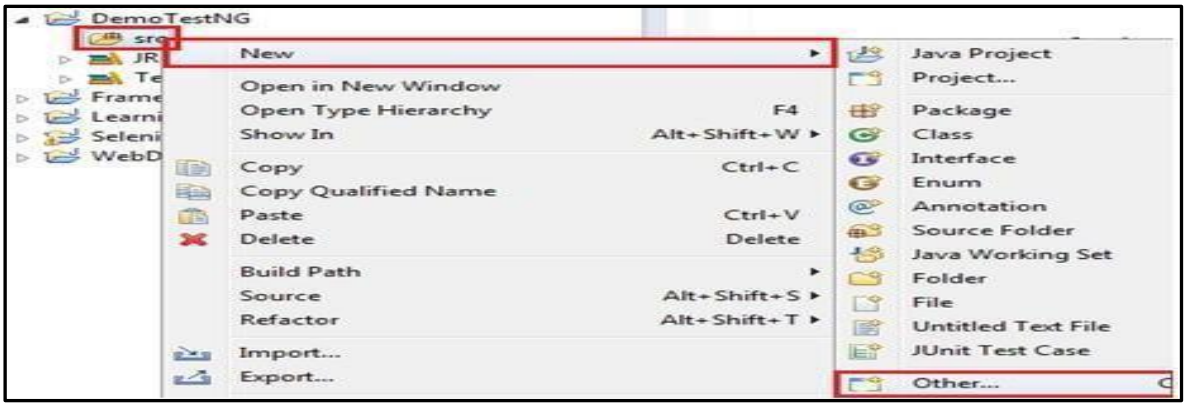


The TestNG is now added to the Java project and the required libraries can be seen in the package explorer upon expanding the project.

Creating TestNG Class:

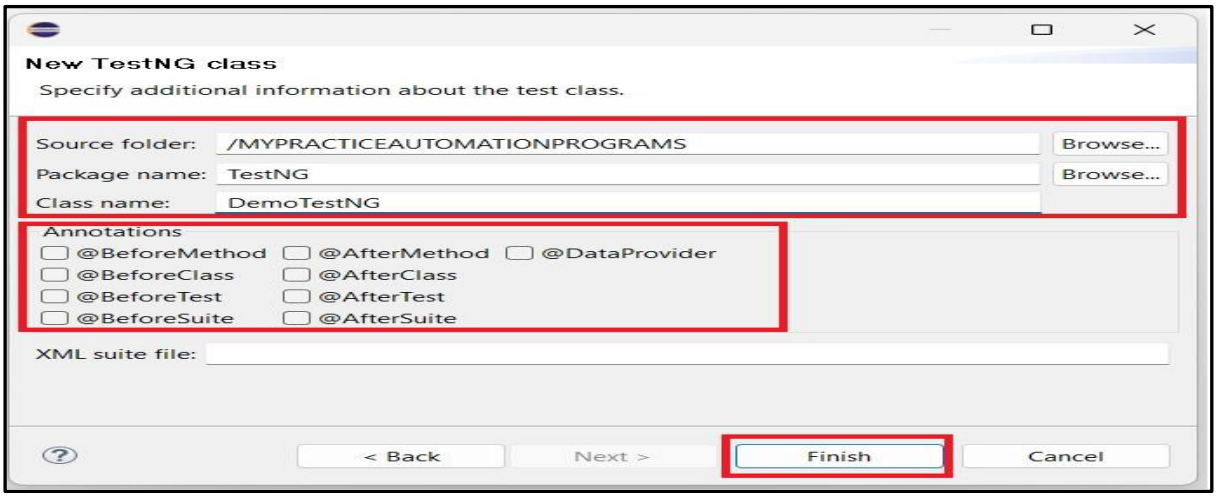
Now that we have done all the basic setup to get started with the test script creation using TestNG. Let's create a sample script using TestNG.

Step 1: Expand the "DemoTestNG" project and traverse to "src" folder. Right-click on the "src" package and navigate to New -> Other.

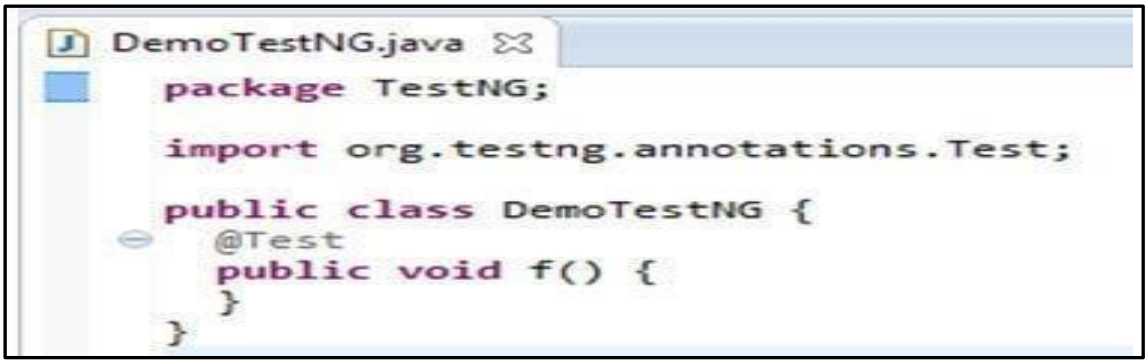


Step 2: Expand TestNG option and select "TestNG" class option and click on the "Next" button.

Step 3: Specify the Source folder, package name and the TestNG class name and click on the Finish button. As it is evident from the below picture, the user can also check various TestNG notations that would be reflected in the test class schema.



The above mentioned TestNG class would be created with the default schema.



Note: **"There is no need of main() method while creating test scripts using TestNG. The program execution is done on the basis of annotations".**

TestNG Annotations:

Annotations in TestNG are lines of code that can control how the method below them will be executed. They are always preceded by the @ symbol. TestNG Annotations are used to control the next method to be executed in the test script. TestNG annotations are defined before every method in the test code. In case any method is not prefixed with annotations, it will be ignored and not be executed as part of the test code. To define them, methods need to be simply annotated with '@ Test' .

✍ Test that your code is multithread safe.

- ✍ Flexible test configuration.
- ✍ Support for data-driven testing (with @DataProvider).
- ✍ Support for parameters.
- ✍ Powerful execution Model (No more TestSuite).
- ✍ Supported by a variety of tools and plug-ins (Eclipse, IDEA, Maven, etc...).
- ✍ Default JDK functions for runtime and logging (no dependencies).

Annotations:

@BeforeSuite: The annotated method will be run before all tests in this suite have run.
@AfterSuite: The annotated method will be run after all tests in this suite have run.
@BeforeTest: The annotated method will be run before any test method belonging to the classes inside the <test> tag is run.
@AfterTest: The annotated method will be run after all the test methods belonging to the classes inside the <test> tag have run.
@BeforeGroups: The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked.
@AfterGroups: The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked.
@BeforeClass: The annotated method will be run before the first test method in the current class is invoked.
@AfterClass: The annotated method will be run after all the test methods in the current class have been run.
@BeforeMethod: The annotated method will be run before each test method.
@AfterMethod: The annotated method will be run after each test method.

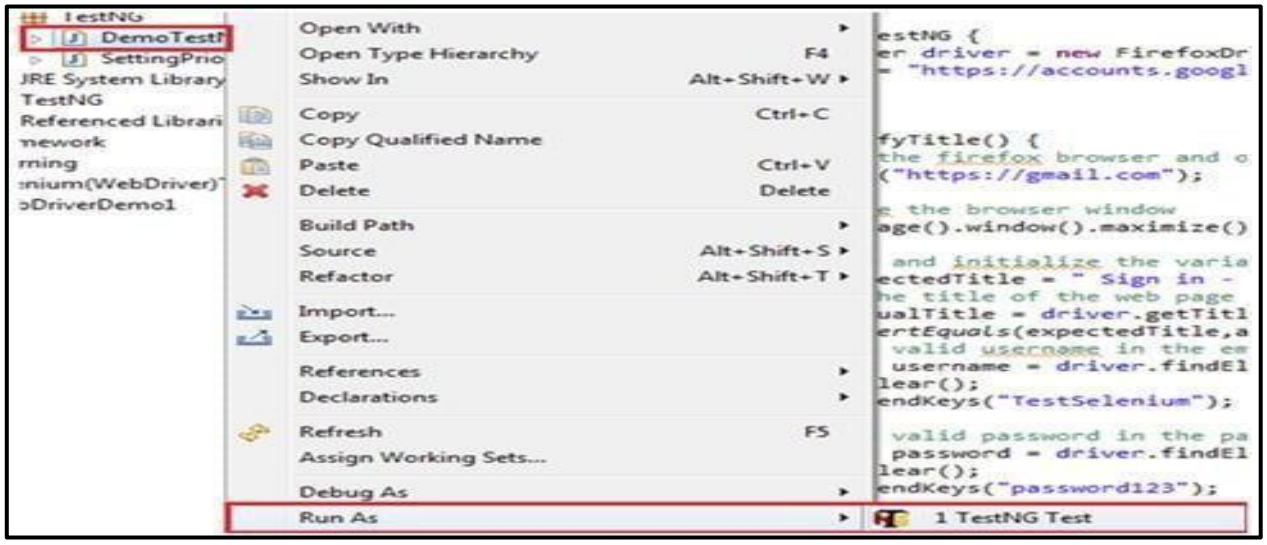
Now that we have created the basic foundation for the TestNG test script, let us now inject the actual test code.

```
package TestNG;
import org.testng.annotations.*
public class TestNGAnnotations {
@BeforeMethod
public void beforeMethod() {
System.out.println(" Before Method will execute before every test method");
}
@AfterMethod
public void afterMethod() {
System.out.println("After Method will execute after every test method ");
}
@BeforeClass
public void beforeClass() {
```

```
System.out.println("Before Class will always execute prior to Before Method and Test Method ");
}
@AfterClass
public void afterClass() {
System.out.println("After Class will always execute later to After Method and Test method");
}
@BeforeTest
public void beforeTest() {
System.out.println("Before Test will always execute prior to Before Class, ,Before Method and Test Method ");
}
@AfterTest
public void afterTest() {
System.out.println("After Test will always execute later to After Method, After Class");
}
@BeforeSuite
public void beforeSuite() {
System.out.println("Before Suite will always execute prior to all annotations or tests in the suite.");
}
@AfterSuite
public void afterSuite() {
System.out.println("After suite will always execute at last when all the annotations or test in the suite have run.");
}
@Test(priority=3) public void testCase1() {
System.out.println("This is my First Test Case 1");
}
@Test(priority=2) public void testCase2() {
System.out.println("This is my Second Test Case 2");
}
@Test(priority=1) public void testCase3() {
System.out.println("This is my Third Test Case 3");
} }
```

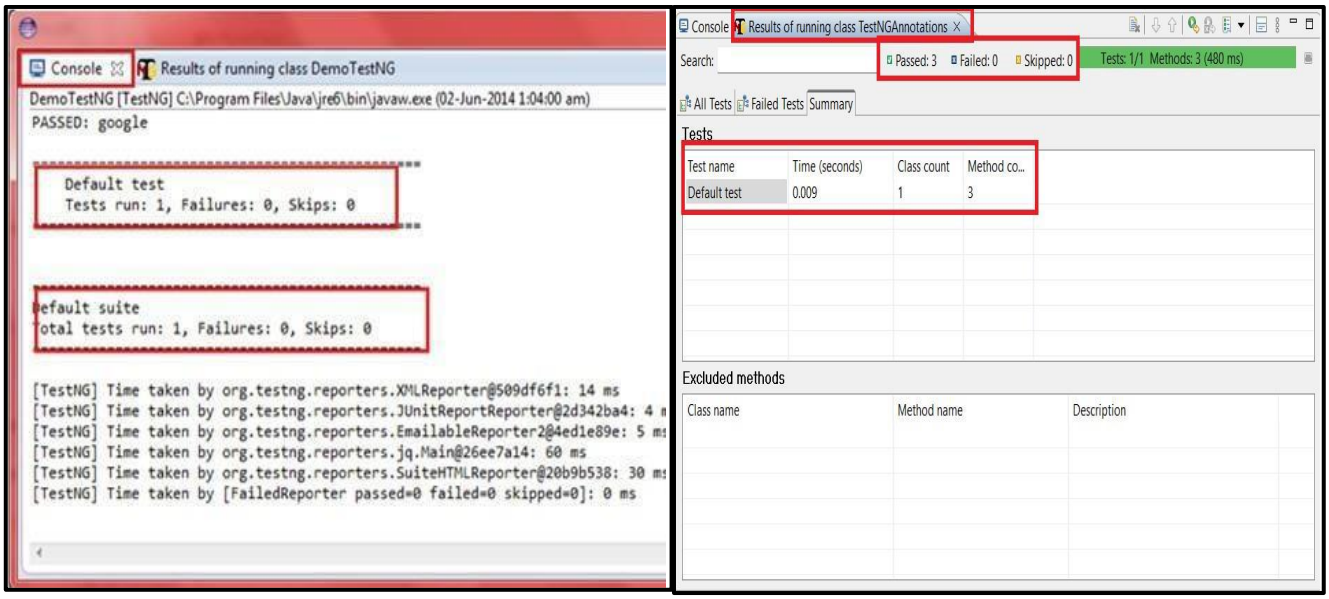
Executing The TestNG Script:

Right click anywhere inside the class within the editor or the java class within the package explorer, select “Run As” option and click on the “TestNG Test”.



TestNG result is displayed into two windows:

→ Console Window **-> TestNG Result Window**



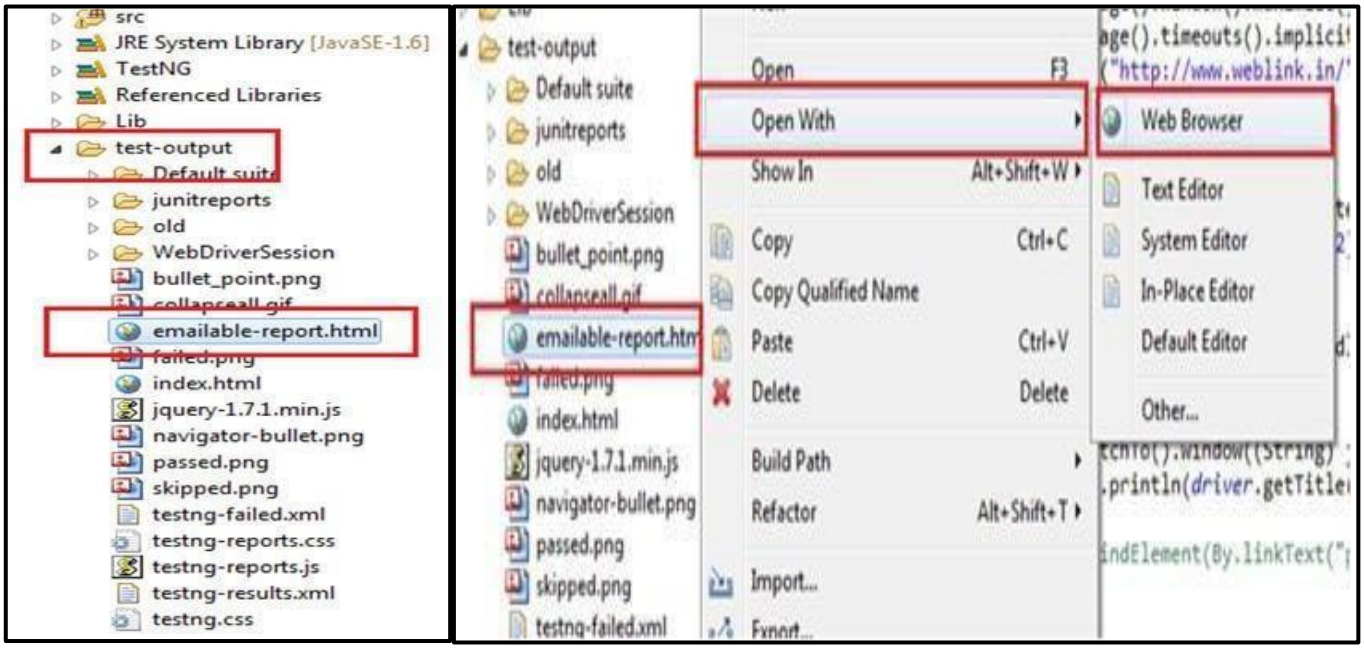
HTML Reports:

TestNG comes with a great capability of generating user readable and comprehensible HTML reports for the test executions. These reports can be viewed in any of the browsers and it can also be viewed using Eclipse's build in browser support.

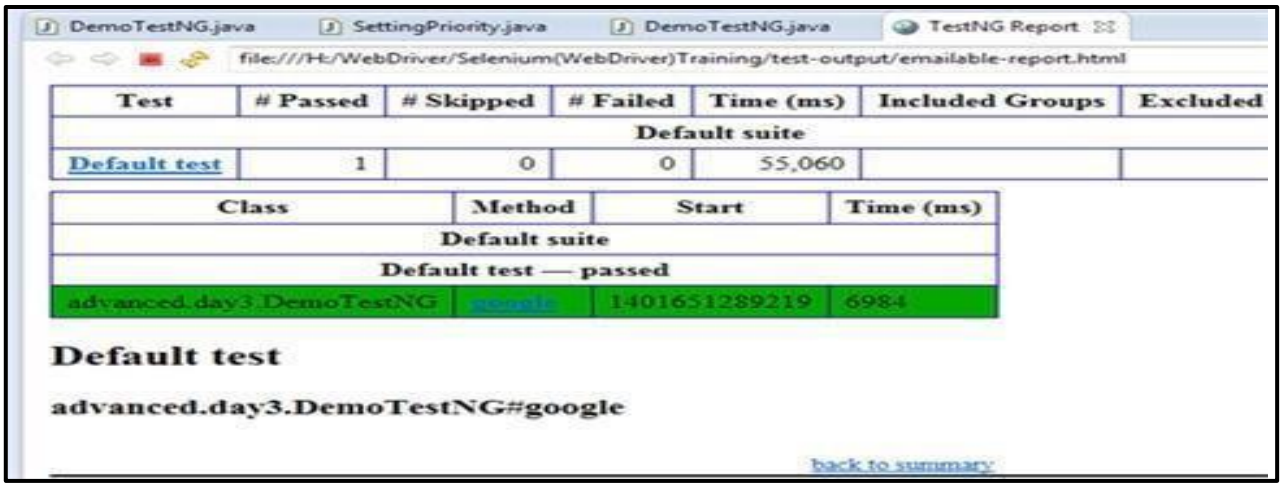
To generate the HTML report, follow the below steps:

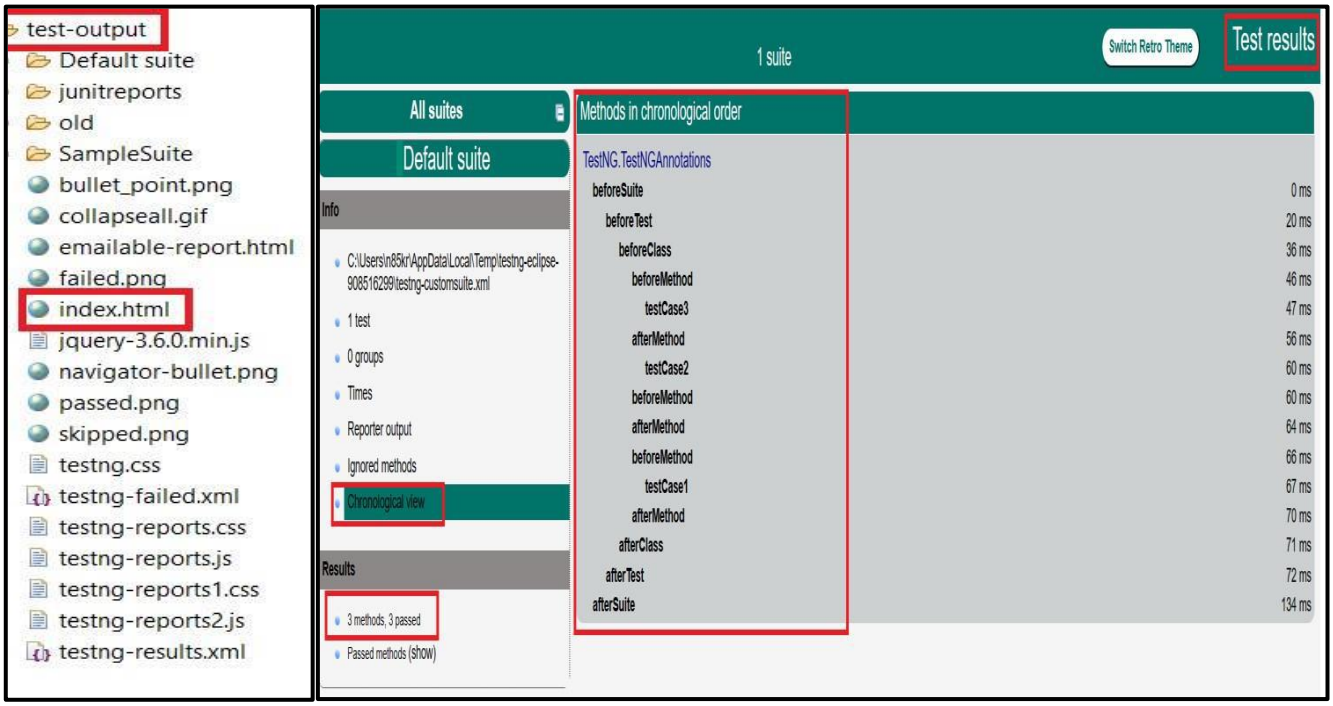
Step 1: Execute the newly created TestNG class. **Refresh** the project containing the TestNG class by right - clicking on it and selecting "Refresh" option.

Step 2: A folder named as "test - output" shall be generated in the project at the "src" folder level. Expand the "test - output" folder and open on the "emailable - report.html" file with the Eclipse browser. The HTML file displays the result of the recent execution.



Step 3: The HTML report shall be opened within the eclipse environment. Refer the below image for the same. Refresh the page to see the results for fresh executions if any.





Setting Priority in TestNG:

If a test script is composed of more than one test method, the execution priority and sequence can be set using TestNG annotation "@Test" and by setting a value for the "priority" parameter.

```
@Test(priority=3)
public void testCase1() {
System.out.println("This is my First Test Case 1");
}

@Test(priority=2)
public void testCase2() {
System.out.println("This is my Second Test Case 2");
}

@Test(priority=1)
public void testCase3() {
System.out.println("This is my Third Test Case 3");
}
```

Code Walkthrough:

In the above code snippet, all the methods are annotated with the help @Test and the priorities are set to 3, 2 and1. Thus the order of execution in which the test methods would be executed is:

TestCase3

TestCase2

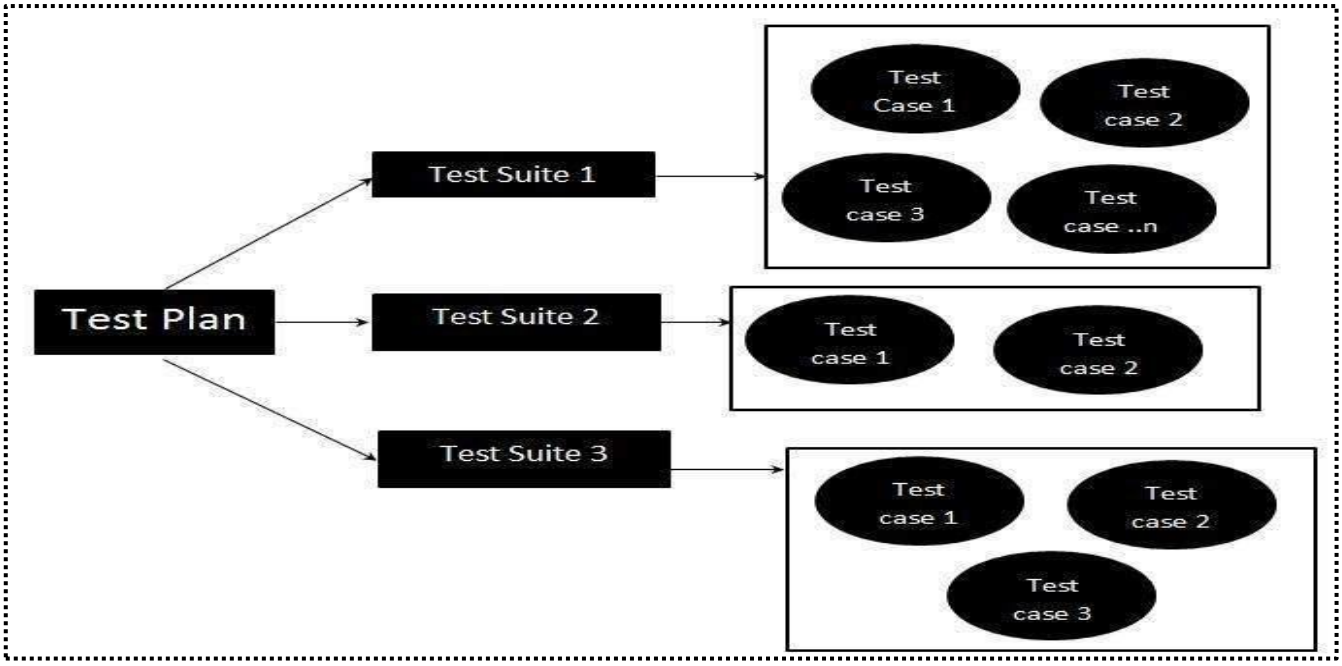
TestCase1

Test Suite:

Suites are used to execute multiple tests together. Suites can be created using both TestNG and JUnit4. However, suites are more powerful in TestNG as it uses very different method for execution of tests. In software development, a test suite, less commonly known as a validation suite, is a collection of test cases that are intended to be used to test a software program to show that it has some specified set of behaviours. A test suite often contains detailed instructions or goals for each collection of test cases and information on the system configuration to be used during testing. A group of test cases may also contain prerequisite states or steps, and descriptions of the following tests. For Example, A system might have a smoke test suite that consists only of smoke tests or a test suite for some specific functionality in the system. It may also contain all tests and signify if a test should be used as a smoke test or for some specific functionality. The suite test refers to group a few unit test cases and run it together. The @RunWith and @Suite annotation are used to run the suite test.

- Test suite is a container that has a set of tests which helps testers in executing and reporting the test execution status. It can take any of the three states namely Active, In progress and completed.

- A Test case can be added to multiple test suites and test plans. After creating a test plan, test suites are created which in turn can have any number of tests.
- Test suites are created based on the cycle or based on the scope. It can contain any type of tests, viz - functional or Non-Functional.



****

Selenium – Java Cheat Sheet

Selenium – Java Cheat Sheet

Driver Initialization

Chrome	WebDriver driver = new ChromeDriver();
Firefox	WebDriver driver = new FirefoxDriver();
Edge	WebDriver driver = new EdgeDriver();
Safari	WebDriver driver = new SafariDriver();

Locating Elements - By:

id:	driver.findElement(By.id("idValue"));
name:	driver.findElement(By.name("nameValue"));
className:	driver.findElement(By.className ("classValue"));
tagName:	driver.findElement(By.tagName ("html tagName"));
cssSelector:	driver.findElement(By.cssSelector("input[type='submit']"));
xPath:	driver.findElement(By.xpath("//input[@type='submit']"));
linkText:	driver.findElement(By.linkText ("Sale"));
partialLinkText:	driver.findElement(By.partialLinkText ("link text"));

Dynamic xPath:

//*[@type='submit']	→ any tag with type submit
//h2[contains(@id, 'ageCont')]	→ selects id that contains ageCont value
(//h2[starts-with(@id, 'u_')])[1]	→ the first input whose id starts with u_
//input[ends-with(@id, 'p7')]	→ selects id that ends with p7
//h2[@id='page-ent' or @class='nav-flex']	→ one or the other statement
//h2[@id='page-ent' and @class='nav-flex']	→ both statements
//*[.= 'Sign in']	→ any tag & attribute just give me the text
//*[(text() = 'Welcome')]	→ selects only text
//*[contains(text(), 'Welcome to')]	→ selects only text that contains
	→ Use index when there are multiple matches

CSS Selector:

.classValue	→ By.cssSelector(".form-control")
#idValue	→ By.cssSelector("#ageCont")

Selenium Operations

Launch a Webpage:

```
driver.get("https://www.google.com");
OR driver.navigate().to("https://www.google.com");
```

Click a button:

```
WebElement searchBtn = driver.findElement(By.name("btnK")).click();
OR searchButton.click();
```

Accept an alert pop-up: `driver.switchTo( ).alert( ).accept();`

Print the page title:

```
String title = driver.getTitle(); System.out.println(title);
```

Clear the input field text:

```
WebElement searchInput = driver.findElement(By.name("q"));
searchInput.sendKeys("selenium"); searchInput.clear();
```

Disable a field (set the 'disabled' attribute):

```
JavascriptExecutor javascript = (JavascriptExecutor) driver;
String toDisable = "document.getElementsByName('fname')[0]
    .setAttribute('disabled', '');";
    javascript.executeScript(toDisable);
```

Enable a field (remove the 'disabled' attribute):

```
JavascriptExecutor javascript = (JavascriptExecutor) driver;
String toEnable = "document.getElementsByName('fname')[0]
    .setAttribute(enabled, '');";
    javascript.executeScript(toEnable);
```

Wait Operations

Selenium Dynamic Wait

Implicit wait – global wait:

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

Explicit wait – local wait:

1. Create WebDriverWait object

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

2. Use the object to add expected conditions

```
WebElement classABC = wait.until(ExpectedConditions
    .visibilityOfElementLocated(By.cssSelector(".classlocator")));
```

→ better than implicit wait when element is not visible / clickable / displayed

FluentWait – local wait. Is like Explicit wait with more options:

```
Wait<WebDriver> fluentWait = new FluentWait<WebDriver>(driver)
    .withTimeout(Duration.ofSeconds(30))
    .pollingEvery(Duration.ofSeconds(5))//will check every 5 sec
    .ignoring(NoSuchElementException.class); //ignores exception
```

Same as Explicit Wait:

```
WebElement classABC = wait.until(ExpectedConditions
    .visibilityOfElementLocated(By.cssSelector(".classlocator")));
```

ScriptTimeout & PageLoad Timeout:

```
driver.manage().timeouts().scriptTimeout(Duration.ofMinutes(2));
driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(10));
```

Java hard wait ->

Sleep: `Thread.sleep(Time in MilliSeconds);`

TestNG Annotations

@Test	the main part of the automation script where we write the business logic we want to automate
@BeforeSuite	runs before executing all test methods in the suite
@BeforeTest	executes before executing all test methods of available classes belonging to that folder
@BeforeClass	executes before the first method of the current class is invoked
@BeforeMethod	executes before each test method runs
@AfterSuite	executes after executing all test methods in the suite
@AfterMethod	executes after executing each test method
@AfterTest	executes after executing all test methods of available classes belonging to that folder
@AfterClass	executes after executing all test methods of the current class

JUnit Annotations

@Test	Represents the method or class as a test block, also accepts parameters.
@Before	The method with this annotation gets executed before all other tests.
@BeforeClass	The method with this annotation gets executed once before class.
@After	The method with this annotation gets executed after all other tests are executed.
@AfterClass	The method with this annotation gets executed once after class.
@Ignore	It is used to ignore certain test statements during execution.
@Disabled	Used to disable the tests from execution, but the corresponding reports of the tests are still generated.

Alerts

Accept an alert: Same as clicking OK of an alert.

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

Dissmiss an alert: Same as clicking Cancel of an alert.

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

Enter text in an alert box:

```
driver.switchTo().alert().sendKeys("Selenium")
```

Retrieve alert text: To get the alert message of the alert.

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

Selenium Navigators

Navigate to a URL	driver.get("URL") OR driver.navigate().to("URL");
Refresh the page	driver.navigate().refresh();
Navigate forward in browser	driver.navigate().forward();
Navigate back in browser	driver.navigate().back();

Java Faker

Copy Faker dependency into pom.xml file

1. Create a Faker object

```
Faker faker = new Faker();
```

2. generate fake data

```
driver
.findElement(By.name("firstname"))
.sendKeys(faker.name().firstName());
```

OR

```
String fName = faker.name().firstName();
```

fake data = mock data → fake ssn, fake name, fake address

Drop Down List

Step 1: Locate the dropdown element:

```
WebElement month=driver.findElement(By.id("dropdown"));
```

Step 2: Create Select object and pass the variable to that object:

```
Select selectMonth=new Select(month);
```

Step 3: Select from a dropdown using select object with 3 different ways:

```
selectMonth.selectByIndex(0);
selectMonth.selectByValue("1");
selectMonth.selectByVisibleText("Jan");
```

We can put all dropdown elements in a List<WebElement> using **getOptions();**

```
Select selectOptions = new Select(states);
List<WebElement> options = selectOptions.getOptions();
```

iFrame

A page within a page → we must first **switch()** to the iframe. 3 ways:

1. by index: → index start from 0

driver.switchTo().frame(0) will switch the first iframe

2. id/name:

```
driver.switchTo().frame("id or name of the iframe");
```

3. web element (locators):

```
WebElement middleFrame =
driver.findElement(By.xpath("//frame[@name='left']"));
driver.switchTo().frame(middleFrame);
```

→ Switching back to parent / default frame:

To parent frame goes only 1 level up:

- `driver.switchTo().parentFrame();`

To get back to the main fraim:

- `driver.switchTo().defaultContent();`

Returns the total number of iframe on a page

- `driver.findElements(By.tagName("iframe"));`

Working with Windows

1. Get the current window handle:

```
String window1Handle = driver.getWindowHandle();
```

2. Get all window handles:

```
Set<String> allWindowHandles = driver.getWindowHandles();
```

3. Switch to a specific window:

```
for (String eachHandle : allWindowHandles){
  if (!eachHandle.equals(window1Handle)){
   driver.switchTo().window(eachHandle);
  }
 }
```

OR

```
String windowHandle = driver.getWindowHandle();
driver.switchTo().window(windowHandle);
```

Switch to newly created window:

```
driver.switchTo().newWindow(WindowType.TAB);
driver.switchTo().newWindow(WindowType.WINDOW);
```

Close the current window:

```
driver.close();
```

Set window position:

```
driver.manage().window().setPosition(new Point(0, 0));
```

Maximize window:

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

Minimize window:

```
driver.manage().window().minimize();
```

Fullscreen window:

```
driver.manage().window().fullscreen();
```

Take a Screenshot:

```
import org.apache.commons.io.FileUtils;
File scrFile = ((TakesScreenshot)driver)
      .getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(scrFile, new File("./image.png"));
```

Actions

Step 1: Create the action object:

```
Actions actions=new Actions(driver);
```

Step 2: Locate the WebElement you want to work on:

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

Step 3: Perform the action on the WebElement

Right click: `actions.contextClick(element).perform();`

Hover over: `actions.moveToElement(element).perform();`

```
actions .sendKeys(Keys.ARROW_DOWN)
        .sendKeys(Keys.ARROW_UP)
        .sendKeys(Keys.PAGE_DOWN)
        .sendKeys(Keys.PAGE_UP)
        .build() //OPTIONAL : recommended with method chains
        .perform(); //MANDATORY
```

keysDown(); → to press and hold a key. Keys mean Shift,Ctrl, Alt keys.

keysUp(); → to release a pressed key after keysDown(), otherwise we may get IllegalArgumentException.

sendKeys(element,"text"); → to type into text box / text area

Working with Files

Upload a file:

```
driver.findElement(By.id("upload")).sendKeys("path/to/the/file.txt");
driver.findElement(By.id("file-submit")).submit();
```

Read data from an Excel file:

<Apache dependancy>

→ workbook > worksheet > row > cell

→ Index starts with 0 → e.g. row 1 cell 1 has the index of row 0 cell 0

1. Store file path in a string

```
String path = "resources/Capitals.xlsx";
```

OR File file = new File("resources/Capitals.xlsx");

2. Open the file

```
FileInputStream fileInputStream = new FileInputStream(path);
```

3. Open the workbook using fileinputstream

```
Workbook workbook = WorkbookFactory.create(fileInputStream);
```

4. Open the first worksheet

```
Sheet sheet1 = workbook.getSheet("Sheet1");
```

OR workbook.getSheetAt(0); //ALTERNATIVE

5. Go to first row

```
Row row1 = sheet1.getRow(0);
```

6. Go to first cell on that first row and print

```
Cell cell1 = row1.getCell(0);
```

Read data from a text file using BufferedReader:

```
FileReader reader = new FileReader("MyFile.txt");
BufferedReader bufferedReader = new BufferedReader(reader);
        String line;
        while ((line = bufferedReader.readLine()) != null)
        { System.out.println(line); }
reader.close();
```

Read data from a text file Using InputStream:

```
FileInputStream inputStream = new FileInputStream("MyFile.txt");
InputStreamReader reader = new InputStreamReader(inputStream,
"UTF-16");
int character;
        while ((character = reader.read()) != -1)
        { System.out.print((char) character); }
reader.close();
```

Read data from a text file Using FileReader:

```
FileReader reader = new FileReader("MyFile.txt");
int character;
        while ((character = reader.read()) != -1)
        { System.out.print((char) character); }
reader.close();
```

Read data from a CSV file:

```
import au.com.bytecode.opencsv.CSVReader;
String path = "C:\\Users\\Myuser\\Desktop\\csvtest.csv";
Reader reader = new FileReader(path);
CSVReader csvreader = new CSVReader(reader);
List<String[]> data = csvreader.readAll();
        for(String[] d : data){
        for(String c : d ){
        System.out.println(c); } }
```

Working with Files

We can't test desktop applications with Selenium. But we can use JAVA

System.getProperty("user.dir"); =>gives the path of the current folder

System.getProperty("user.home"); =>gives you the user folder

Files.exists(Paths.get("path of the file"));

=>Checks if a file path exists on your computer or not

Javascript Executor

1. Creating a reference

```
JavascriptExecutor js = (JavascriptExecutor) driver;
```

2. Calling the method

```
js.exectueScript(Script, Arguments);
js.executeScript(return something);
```

Example: Clicking on a button

```
WebElement button =driver.findElement(By.name("btnLogin"));
```

//Perform Click on LOGIN button using JavascriptExecutor

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

//arguments[0] -> the first argument in executeScript method

Selenium Grid

Start the hub:

```
java -jar selenium-server-standalone-x.y.z.jar -role hub
```

Start a node:

```
java -jar selenium-server-standalone-x.y.z.jar -role node -hub
```

Server

```
http://localhost:4444/ui/index.html
```


****

Selenium Maven Project with Eclipse

Maven is a build automation tool that is used to manage the project dependency and the whole project lifecycle. Maven is built by Apache Software Foundation and is used majorly for Java projects. Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information. Maven can be more specifically defined as a software project management tool that uses the concepts of the project object model (POM). It enables the user to create an initial folder structure, perform compilation and testing and then package and deploy the final product. It is basically used to manage the life cycle of a project. Maven makes the build management process much easier, as you'll only need to specify the dependencies in the pom.xml files and Maven will take care of the rest!. Maven is a part of the Apache Software Foundation.

Maven is a build automation tool. Maven is a Yiddish word which means "accumulator of knowledge". It was first started by the Apache Software Foundation (ASF) in the year 2002 for the Jakarta Alexandria project.

Some of the key reasons Maven is used are:

- ✓ Maven is a powerful project management tool that is based on the POM (project object model).
- ✓ It simplifies the build process & provides a uniform system
- ✓ It handles compilation, distribution, dependency management and other tasks efficiently.
- ✓ It increases reusability.
- ✓ Also has the ability to work with multiple projects at the same time.
- ✓ It reduces the steps like adding jar files to the project library, building reports, executing Junit test cases, creating jar/war/ear files for deployment.
- ✓ It has a repository that is centralized that manages jar files. Maven reads the pom.xml file.

Maven Pom Xml File [POM]:

POM refers to Project Object Model. It is an XML file which contains the information about the project and various configuration detail used by Maven to build the project like build directory, source directory, dependencies, test source directory, plugin, goals etc. Before maven 2, it was named as project.xml file. But, since maven 2 (also in maven 3), it is renamed as pom.xml.

Elements of maven pom.xml file:

Element	Description
project	It is the root element of pom.xml file.
modelVersion	It is the sub element of project. It specifies the modelVersion. It should be set to 4.0.0.
groupId	It is the sub element of project. It specifies the id for the project group.
artifactId	It is the sub element of project. It specifies the id for the artifact (project). An artifact is something that is either produced or used by a project. Examples of artifacts produced by Maven for a project include: JARs, source and binary distributions, and WARs.
version	It is the sub element of project. It specifies the version of the artifact under given group.

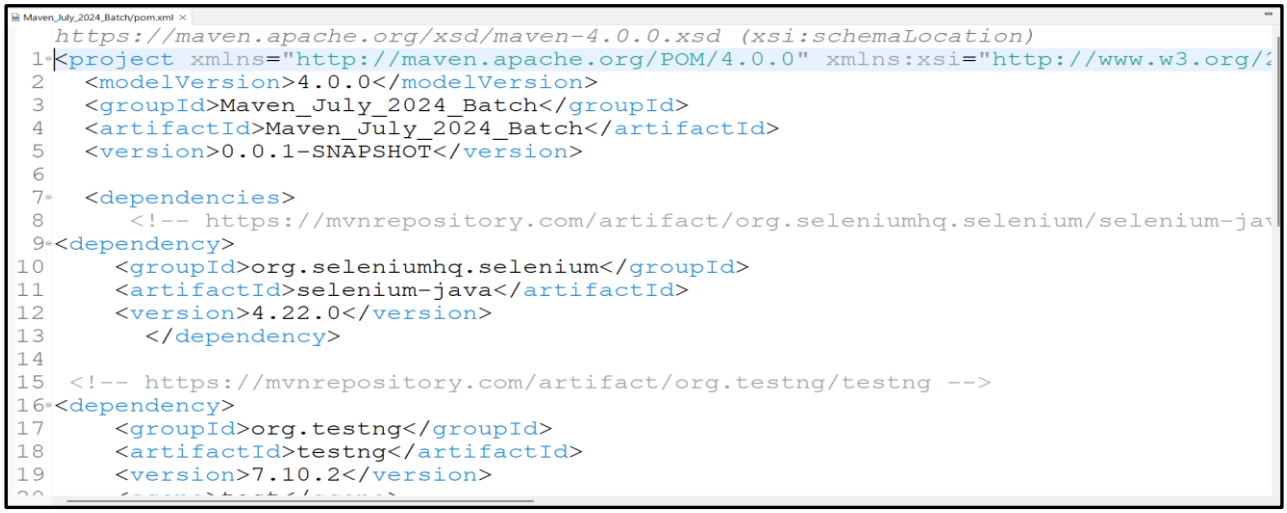
We consider using the Eclipse platform to work with Selenium Maven because Eclipse IDE is the most popular editor to develop Java applications and it is free and easy to understand and has more community support.

Maven Dependency:

In Maven, a dependency is just another archive—JAR, ZIP, and so on—which our current project needs in order to compile, build, test, and/or run. These project dependencies are collectively specified in the pom.xml file, inside of a <dependencies> tag. When we run a maven build or execute a maven goal, these dependencies are resolved and then loaded from the local repository. If these dependencies are not present in the local repository, then Maven will download them from a remote repository and cache them in the local repository.

For solving external libraries dependency problem Maven provides dependency management functionality for adding external libraries dependency to the project by adding some configurations into maven pom.xml file of project and then Maven downloads them for you and puts them in your local Maven repository. If any of these external libraries need other libraries, then these other libraries are also downloaded into your local Maven repository.

Maven Dependency Example:



****

Test Automation Framework

A framework is just a set of rules and structures, which makes it easy to get a suite of tests up and testing in minimal time. For example, assign test data in a specific folder, store configuration settings in a specific file and folder, name the tests in this template, create packages, etc. In general, a framework is a real or conceptual structure intended to serve as a support or guide for the building of something that expands the structure into something useful.

Why do we need Automation framework?

- Writing code once and reusing it.
- Significant Reduction in Testing Cycle Time
- Running the script with different set of data.
- Executing the scripts end-to-end without any manual intervention. (If any error occurs from tool or application, Script run will stop. If we use framework, it will skip or fail that testcase and run with the next testcase.)
- With basic knowledge on tool also anyone can run and write the script. (All the script, Keywords has been written by experts, we have to know how to use those keywords)
- Can able to run scenarios by selecting YES or NO. (Modular Driven Framework)

Features & Capabilities of the Framework:

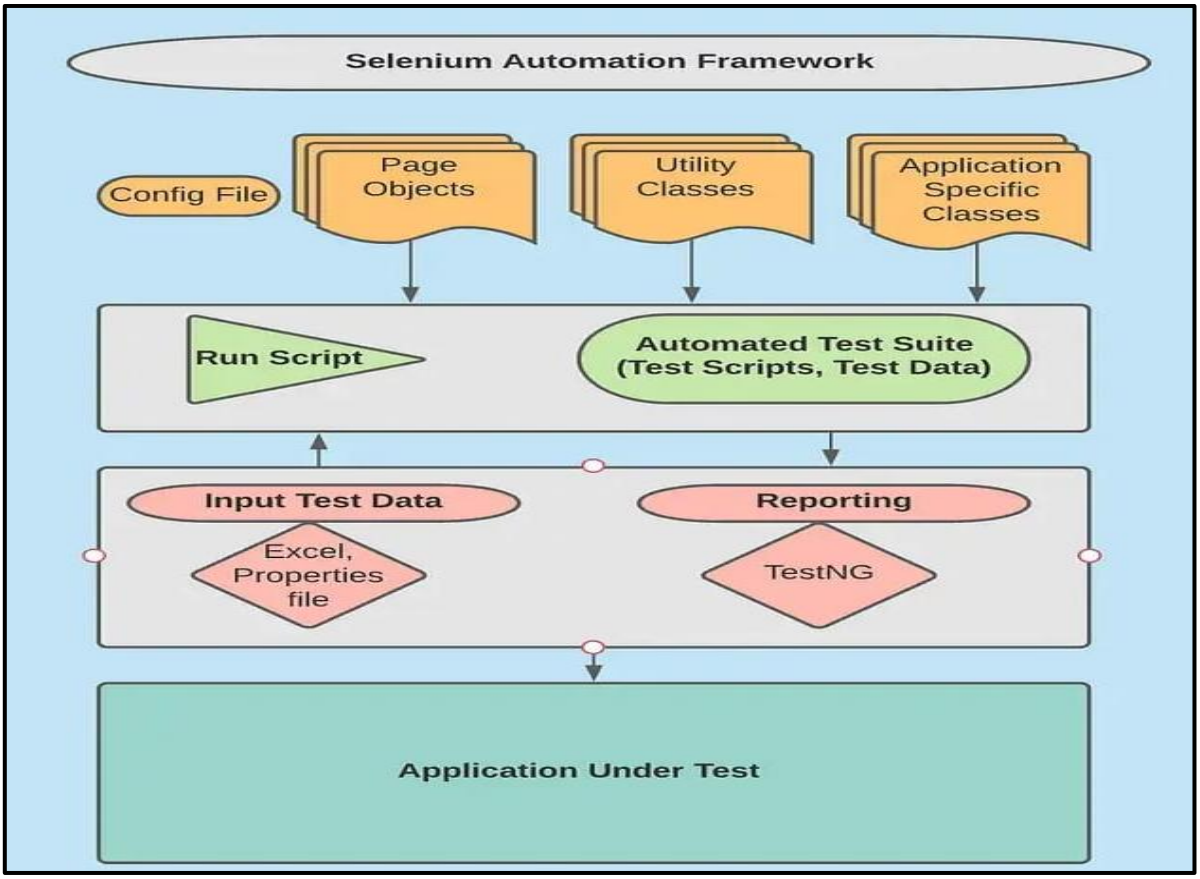
1. **Java** – It uses Java programming language.
2. **TestNG** – It uses TestNG as a testing framework. You can learn more about TestNG from our – TestNG tutorial.
3. **Maven-Based** – It will be maven-based. So all the dependencies will be in a POM file and the test suite can be triggered using maven commands.
4. **Hybrid Framework** – It will be a hybrid framework with a combination of the modular and data-driven framework.
5. **Page Object Model** – The framework will use the Page Object Model design pattern in Selenium.
6. **Page Factory** – It will use page factory implementation of the page object model in Selenium.
7. **Screenshot on failure** – The framework will have the capability to capture screenshots in case of failed tests.
8. **Test data in Excel** – The framework will have a utility class that will read test data from an excel file.
9. **Logging** – Log4j is used for logging.

Benefits of Test Automation:

- It reduces the time to do tests, thus saving manual effort
- It runs tests parallelly, thus reducing manual effort for testing
- It runs tests with more than one browser, thus saving manual effort
- It runs more tests with different sets of data, thus getting better coverage

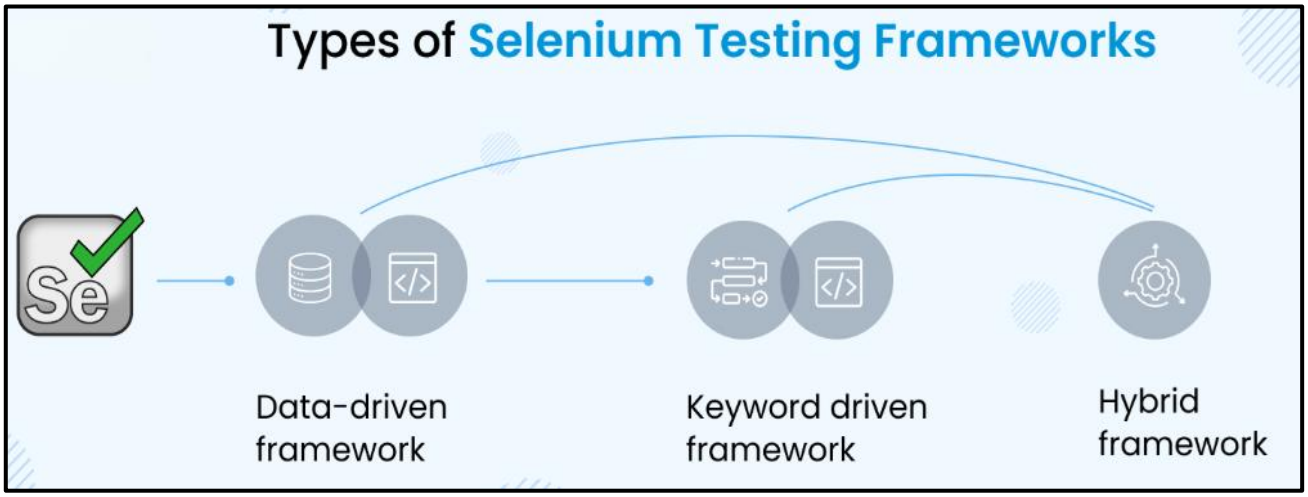
Selenium Automation Framework

The Selenium automation Framework is a code structure that makes code maintenance easy and efficient. Without frameworks, users may place the “code” and “data” at the same location which is neither reusable nor readable. Selenium framework is a code structure for making code maintenance simpler, and code readability better. A framework involves breaking the entire code into smaller pieces of code, which test a particular functionality. Frameworks produce beneficial outcomes like increased code reusability, higher portability, reduced cost of script maintenance, better code readability, etc. Selenium framework is a code structure for making code maintenance simpler, and code readability better. A framework involves breaking the entire code into smaller pieces of code, which test a particular functionality.



Types of Selenium Framework: There are mainly three type of frameworks created by Selenium WebDriver to automate manual test cases.

I. **Data Driven Framework**
II. **Keyword Driven Framework**
III. **Hybrid Driven Framework**



Hybrid Driven Framework is a combination of both the Data-Driven and Keyword-Driven framework. Here, the keywords, as well as the test data, are externalized. Keywords are maintained in a separate Java class file and test data can be maintained either in a properties file/excel file/can use the data provider of a TestNG framework.

Steps to create the framework in Eclipse:

Let us see how to create a new framework in Eclipse

- ✓ Create a new Maven Project
- ✓ Provide a suitable package name for your project :: com… is a good naming convention you can follow
- ✓ The folder structure would look like this: src/main/java, src/main/resources, src/test/java and src/test/resources

All our code would code into two parts.

- ✓ Test Objects and Data Providers would go into src/test/java :: Meaning all testng related code should go into src/test/java
- ✓ Page Objects should go into /src/main/java
- ✓ You would have named your package when creating the maven package structure
- ✓ Within the src/main/java and src/test/java you would see your defined package name and a sample java file will be found. Remove these files.

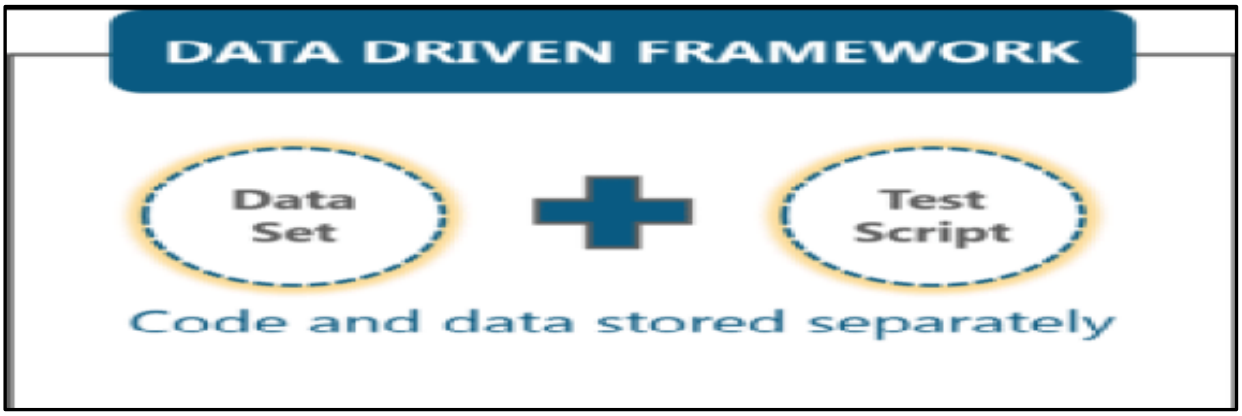
Your project structure should look something like below.

- ✓ Under src/main/java you can have com….pages
- ✓ Under src/test/java you can have com….tests and com….datastore
- ✓ You can have sub packages within these main packages to identify with modules

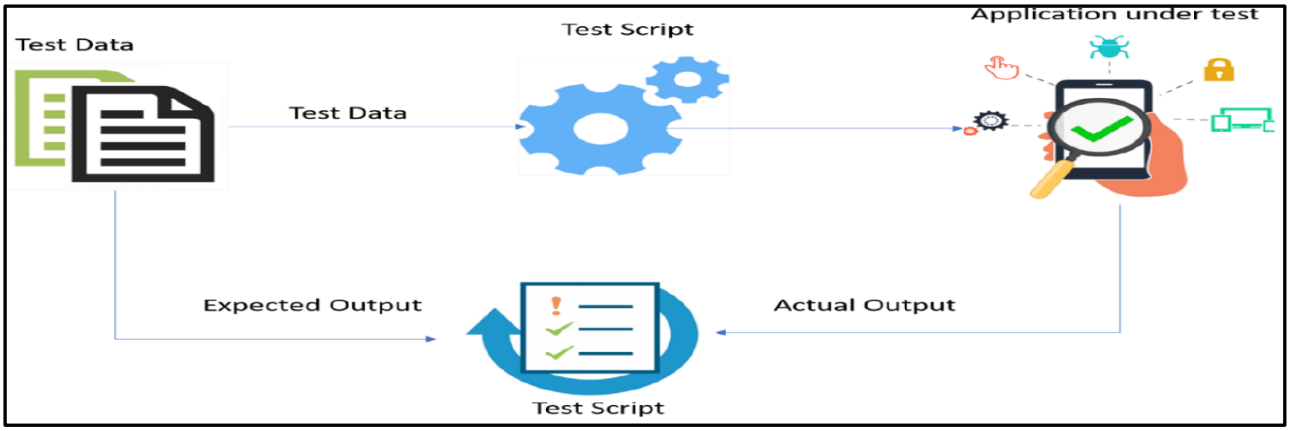
****

Data Driven Framework

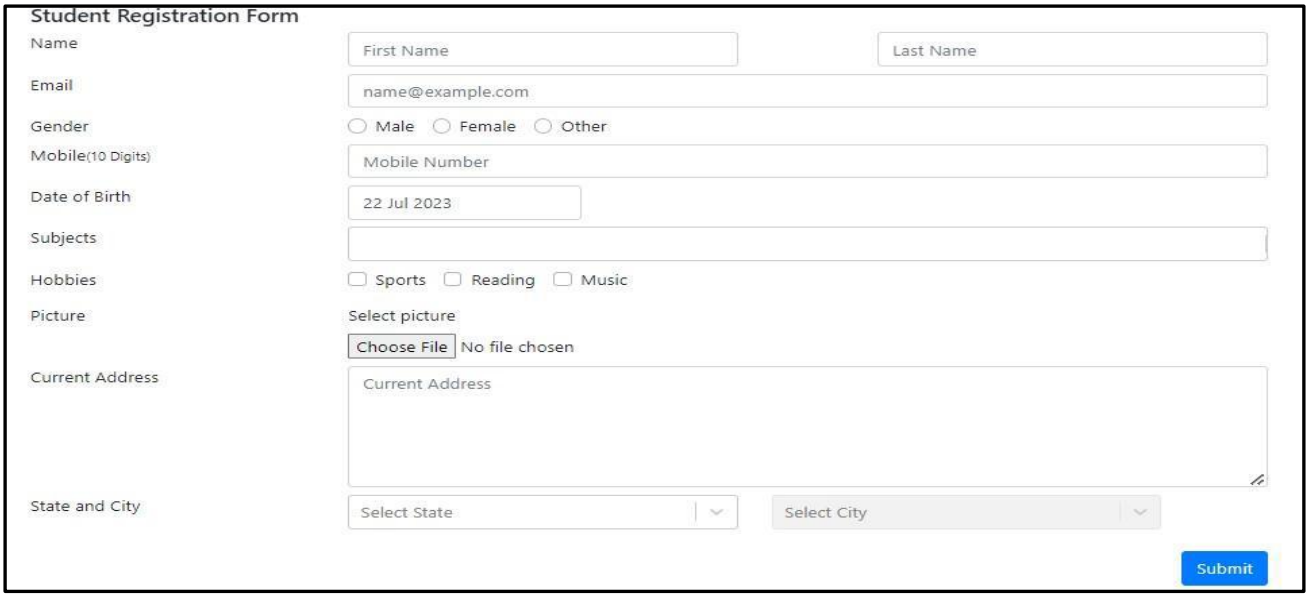
Data Driven Framework in Selenium is a method of separating data sets from the test case. Once the data sets are separated from the test case, it can be easily modified for a specific functionality without changing the code. It is used to fetch test cases and suites from external files like Excel, .csv, .xml or some database tables. Since the test case is separated from the data set, we can easily modify the test case of a particular functionality without making wholesale changes to your code. For example, if you want to modify the code for login functionality, then you can modify just that instead of having to also modify any other dependent portion in the same code.



Data Driven framework is used to drive test cases and suites from an external data feed. The data feed can be data sheets like xls, xlsx, and csv files. A Data Driven Framework in Selenium is a technique of separating the "data set" from the actual "test case" (code). Since the test case is separated from the data set, one can easily modify the test case of a particular functionality without making changes to the code. Data Driven framework is focused on separating the test scripts logic and the test data from each other. Allows us to create test automation scripts by passing different sets of test data. The test data set is kept in the external files or resources such as MS Excel Sheets, MS Access Tables, SQL Database, XML files etc., The test scripts connect to the external resources to get the test data. Data Driven Framework is one of the popular Automation Testing Framework in the current market. Data Driven automated testing is a method in which the test data set is created in the excel sheet, and is then imported into automation testing tools to feed to the software under test.

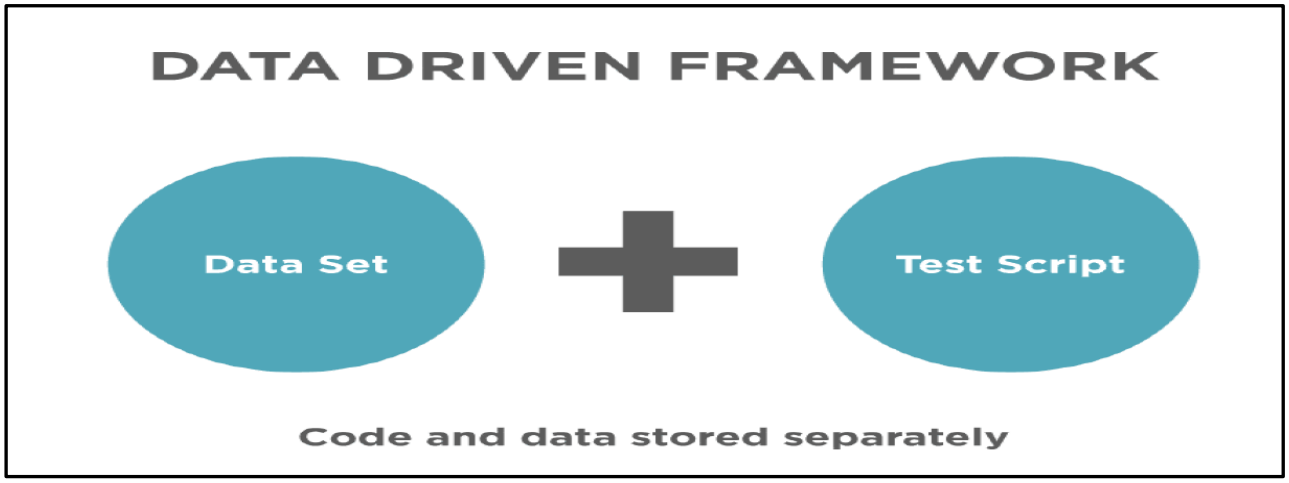


Example: Write an automation script to fill the **Student Registration Form**. There can be many students for registration, the only thing that is differentiating in the code is input values (Name, Address, Phone, Gender, etc..). Will you write a separate script for every student to register? Is there a way, we can reuse the code and only change the student data? Yes, this is where the Data Driven Framework comes into play and makes the test scripts work properly for different sets of test data. It saves time to write additional code. It's like write once and run many times mechanism as you can run the same Selenium script multiple times. In simple words, we use the Data Driven Framework when we have to execute the same script with multiple sets of test data, whose storage is at a different place and not present inside the test script. Any changes done to the data will not impact the code of the test. One of the most commonly used, data sources for the test is Microsoft Excel Sheets.



Benefits of using the Data Driven Testing Framework:

- Test cases can be modified without much changes to code.
- It allows testing the application with multiple sets of data values, especially during regression testing.
- It helps us to separate the logic of the test cases/scripts from the test data.



WebDriver does not directly support data reading of excel files. Therefore, one needs to use third party APIs like Apache POI for reading/writing on any Microsoft office document.

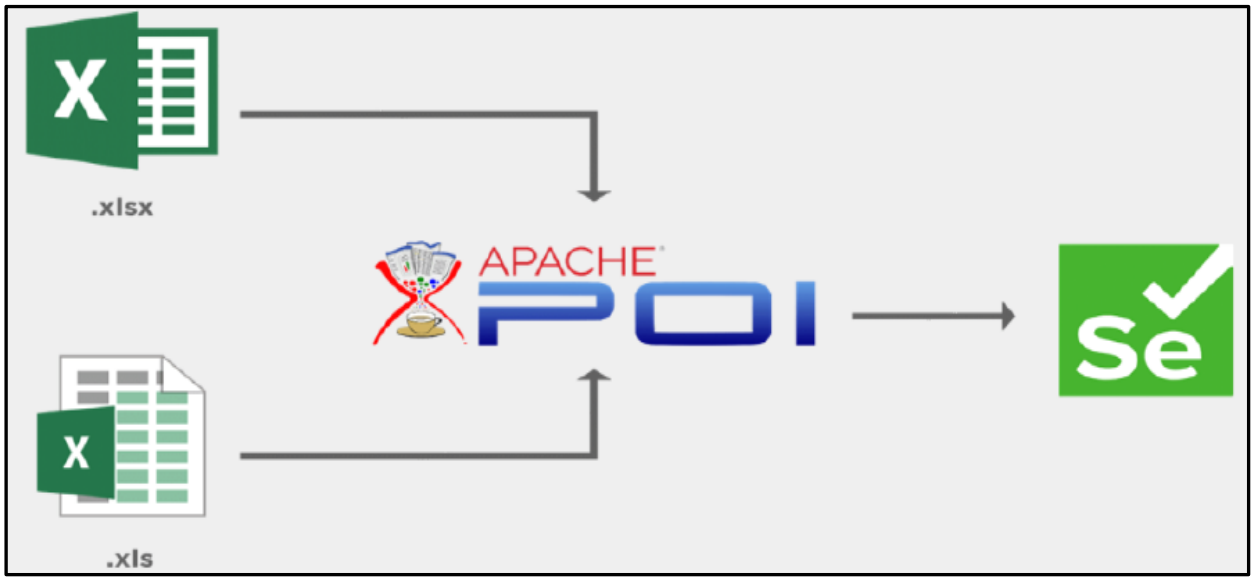
Read and Write Data from Excel Files in Selenium: [Apache POI]

The Apache POI in Selenium is a widely used API for Selenium Data-Driven Testing. Selenium WebDriver is the most used automation tool for the automation of web applications. Now, we know that these web applications are used by multiple users, and each one of those uses the applications as per their own data. So, considering the usage, it becomes the primary responsibility of the QAs also to test the web applications with varying data sets.

Now the user journeys will be the same, but the data set will be different. Therefore, it makes more sense to execute the same test case with different data, instead of writing a separate test case for each user journey with each data set. This is where Microsoft Excel comes in handy, which is one of the favourite tools for storing test data. Excel in Selenium is one of the most used combinations for storing test data and then running the same test case against various data sets. There are various libraries in JAVA which helps in reading/writing data from Excel files. But Apache POI is one of the most used libraries, which provides various classes and methods to read/write data from various formats of Excel files (xls, xlsx etc). Here, we will understand the details of Apache POI and how we can use the same to read/write data from Excel files.

Apache POI:

The Apache POI in Selenium is a widely used API for Selenium Data-Driven Testing. Apache POI, where POI stands for (Poor Obfuscation Implementation) is an API that offers a collection of Java libraries that helps us to read, write, and manipulate different Microsoft files such as excel sheets, power-point, and word files. Apache POI is an open-source Java library often utilized to create and handle Microsoft Office-based files. Since Java does not offer built-in support for Excel files, testers need open-source APIs to work with them. Apache POI provides a Java API that lets users operate.



The Apache POI in Selenium is a widely used API for selenium data driven testing. It is a POI library written in Java that gives users an API for manipulating Microsoft documents like .xls and .xlsx. Users can easily create, modify and read/write into excel files.

Apache POI uses certain terms to work with Microsoft Excel:

Term	Details
Workbook	A workbook represents a Microsoft Excel file. It can be used for creating and maintaining the spreadsheet. A workbook may contain many sheets.
Sheet	A sheet refers to a page in a Microsoft Excel file that contains the number of rows and columns. [XSSFSheet and HSSFSheet classes implement this interface.]
Row	A row represents a collection of cells, which is used to represent a row in the spreadsheet [XSSFRow and HSSFRow classes implement this interface.]
Cell	A cell is indicated by a row and column combination. Data entered by a user is stored in a cell. Data can be of the type such as string, numeric value, or formula. [XSSFCell and HSSFCell classes implement this interface]

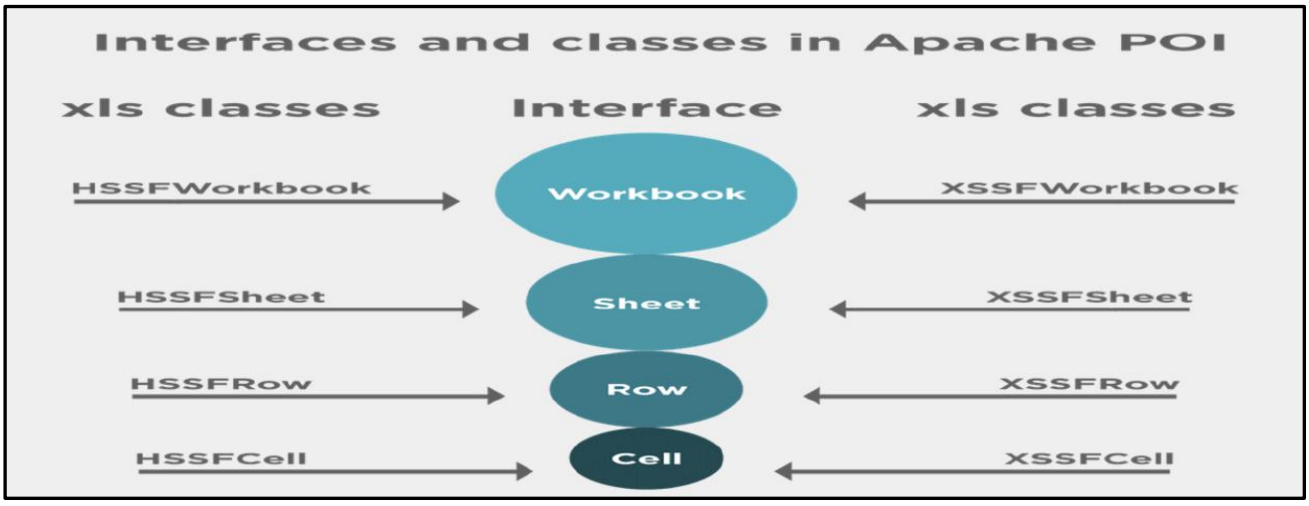
Apache POI have different classes and method to work upon different MS Office Document. Apache POI libraries are used to perform such operations. Some of the interfaces to read or write data from XLS and XLSX file external resources are given below.

"If you are using MS-Office versions 97–2003 use HSSFWorkbook. MS-Office versions 2007 or later use XSSFWorkbook."

Component	Explanation	Details
POIFS	Poor Obfuscation Implementation File System.	This component is the basic factor of all other POI elements, used to read different files.
HSSF	Horrible Spreadsheet Format	This component is used to read/write an older format of Excel(xls).
XSSF	XML Spreadsheet Format	This component is used to read/write a new format of Excel(xlsx).
XSSFWorkbook	XML Spreadsheet Format Workbook	This class representation will implement Workbook interfaces for the XLSX file.
HSSFWorkbook	Horrible Spreadsheet Format Work Book	This class representation will implement the Workbook interface for the XLS file.
XSSFSheet	XML Spreadsheet Format Sheet	This class representing a Sheet interface for the XLSX file.
HSSFSheet	Horrible Spreadsheet Format Sheet	This class representing a Sheet interface for the XLS file.
XSSFRow	XML Spreadsheet Format Row	This class representing a Row interface for the XLSX file.

HSSFRow	Horrible Spreadsheet Format Row	This class representing a Row interface for the XLS file.
XSSFCell	XML Spreadsheet Format Cell	This class representing a Cell interface for the XLSX file.
HWPF	Horrible Word Processor Format	This component reads/writes an older format of Word(doc).
XWPF	XML Word Processor Format	This component reads/writes a new format of Word(docx).
HSLF	Horrible Slide Layout Format	This component reads/writes PowerPoint presentations.

The below image clearly depicts the structure and how the classes and interfaces are aligned in Apache POI.



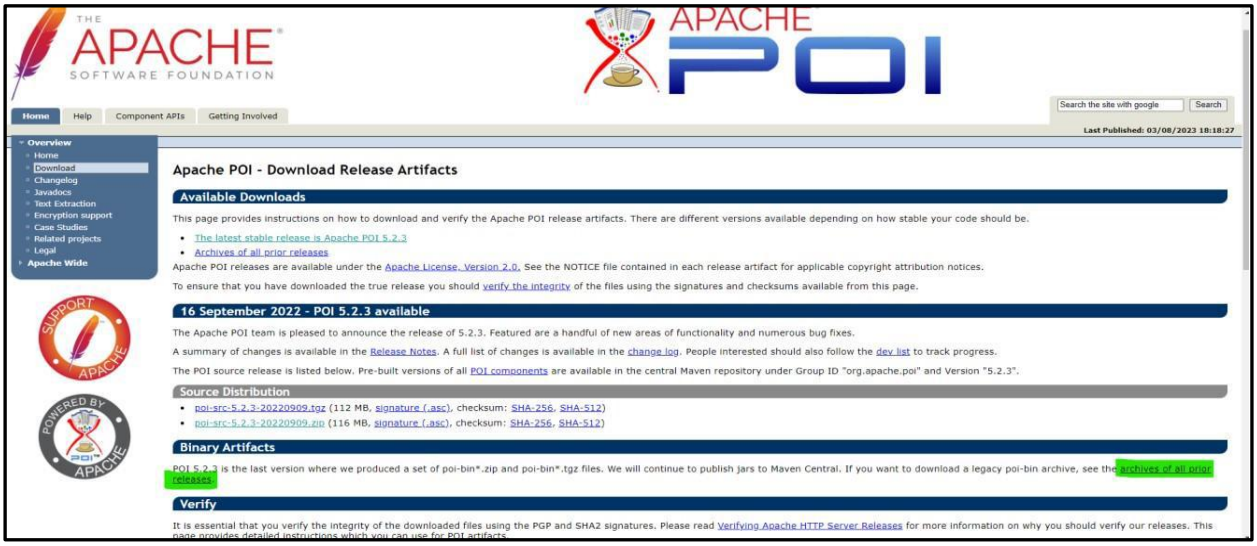
Apache POI Installation:

Apache POI (Poor Obfuscation Implementation) is the Java API for the Microsoft documentation which allows to write or read files in Microsoft formats i.e word, power point & excel. It is an open-source library developed & distributed by Apache software foundations to design or modify Microsoft Office files using java programs.

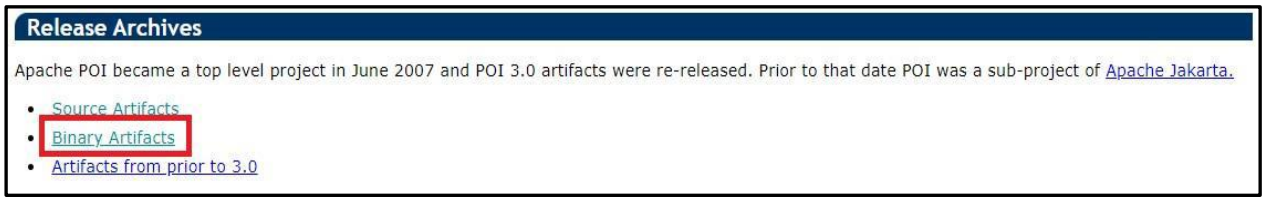
These libraries are used in Selenium to perform read or write operation to the excel(xls or xlsx) files.

How to download Apache POI:

Step1: Download the Apache POI jar file from the official website and click on the Download section. To download the Apache POI Libraries, redirect to the following link: https://poi.apache.org/download.html , under the "Binary Artifacts" click on the "archives of all prior releases".



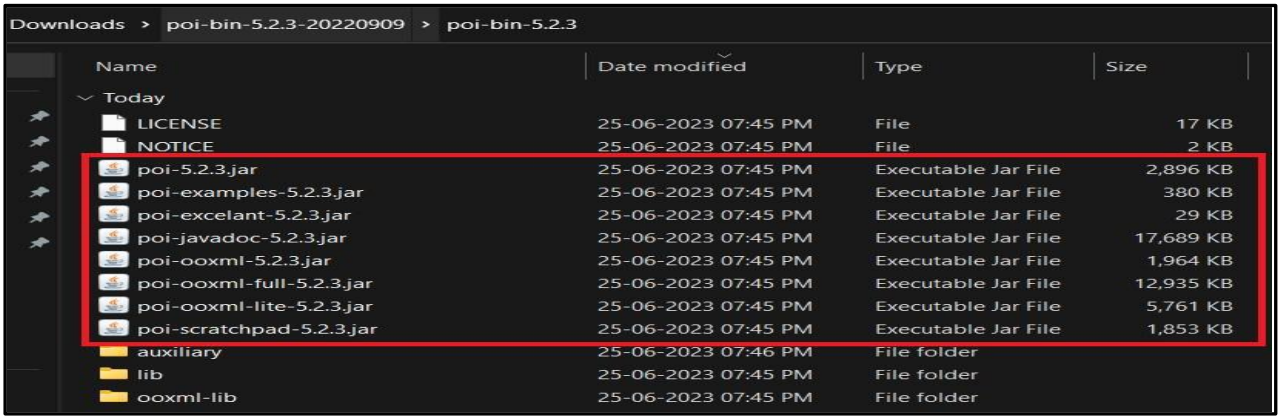
Step2: Click on Binary Artifacts

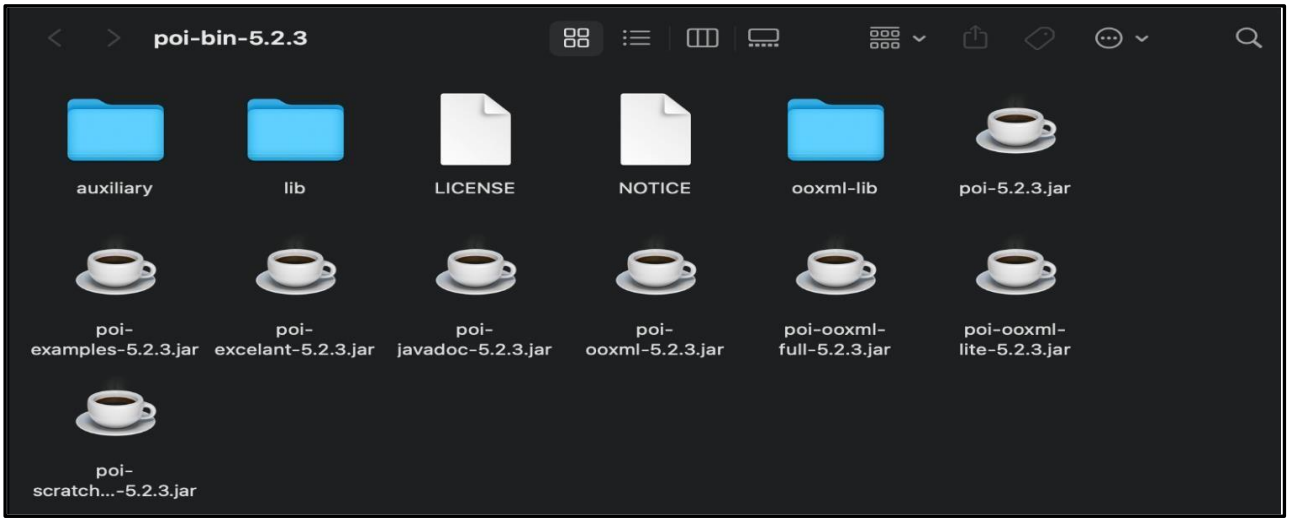


Step3: Under the "Binary Artifacts" click on the "poi-bin-5.2.3-20220909.zip".

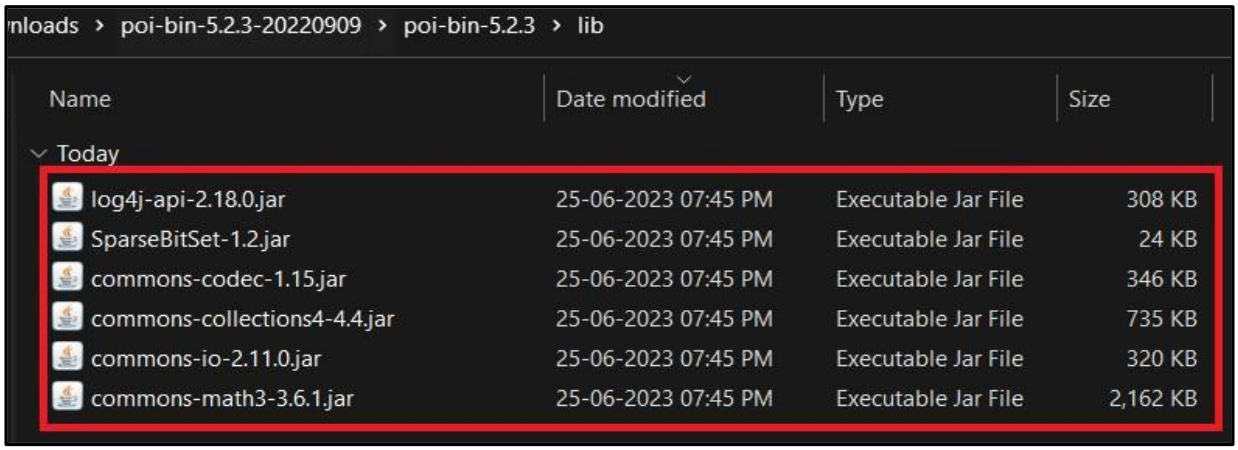
Step4: Extract the downloaded file i.e poi-bin-5.2.3(latest available version) & check for the jar files which are used for your project.

Under the "poi-bin-5.2.3" folder, the following jar files are found:

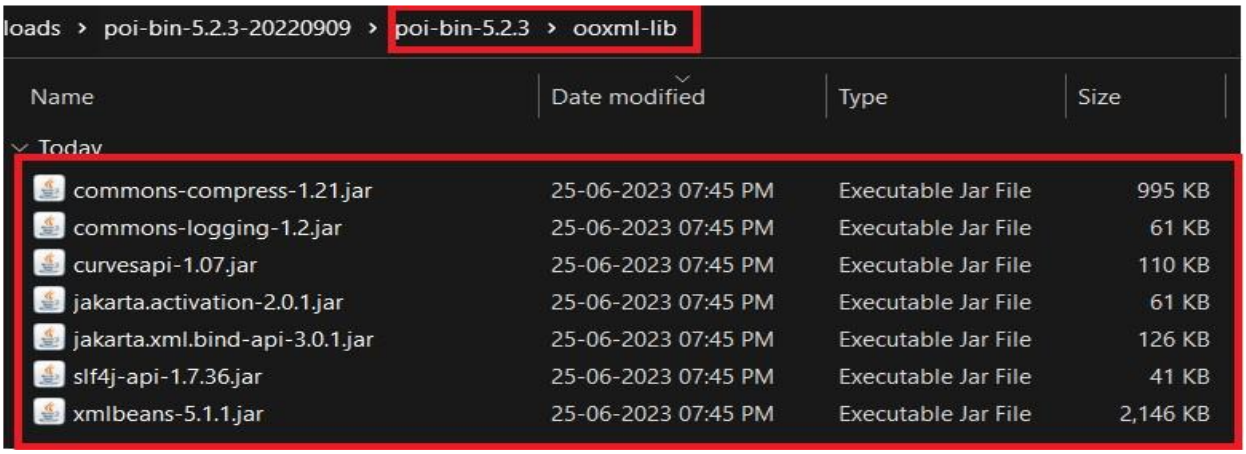




Step5: Under the "poi-bin-5.2.3\lib" folder, the following jar files are found:

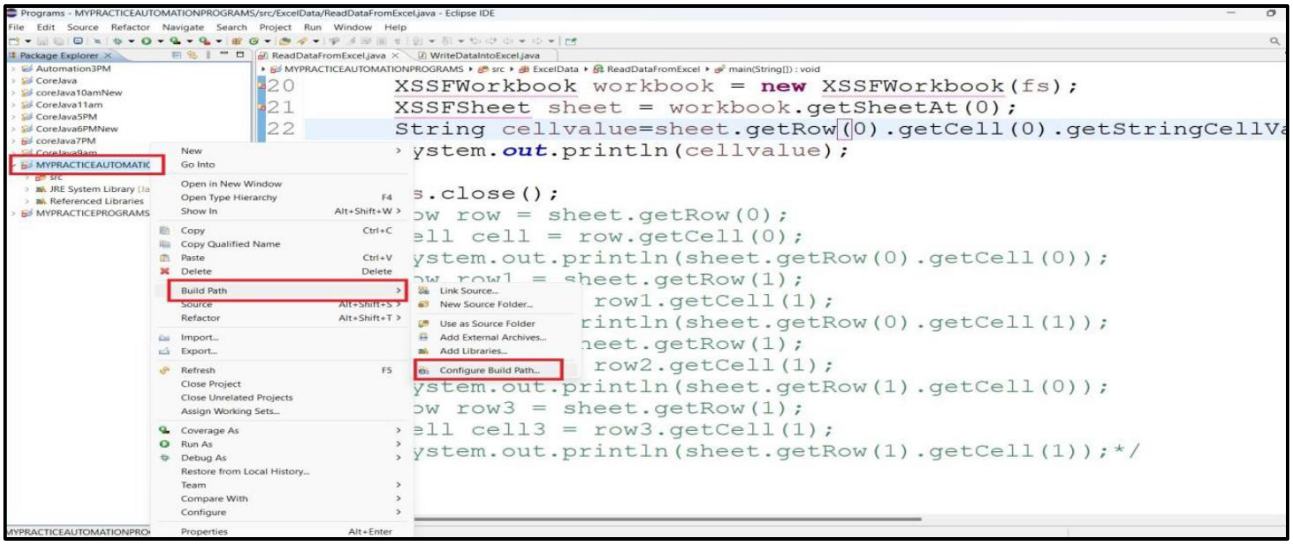


Step6: Under the "poi-bin-5.2.3\ooxml-lib" folder, the following jar files are found:

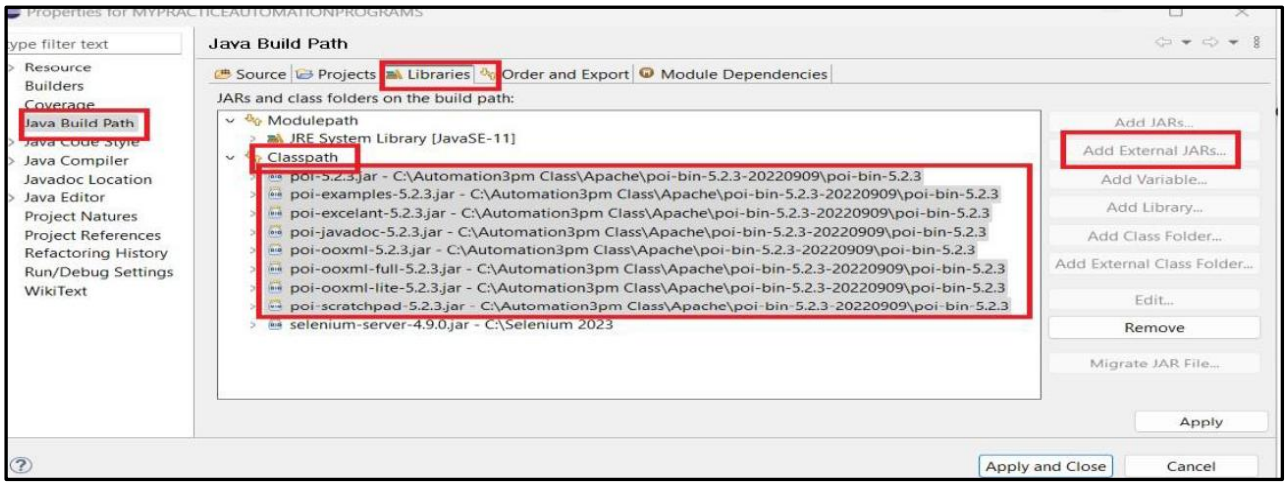


How to configure POI libraries in Eclipse:

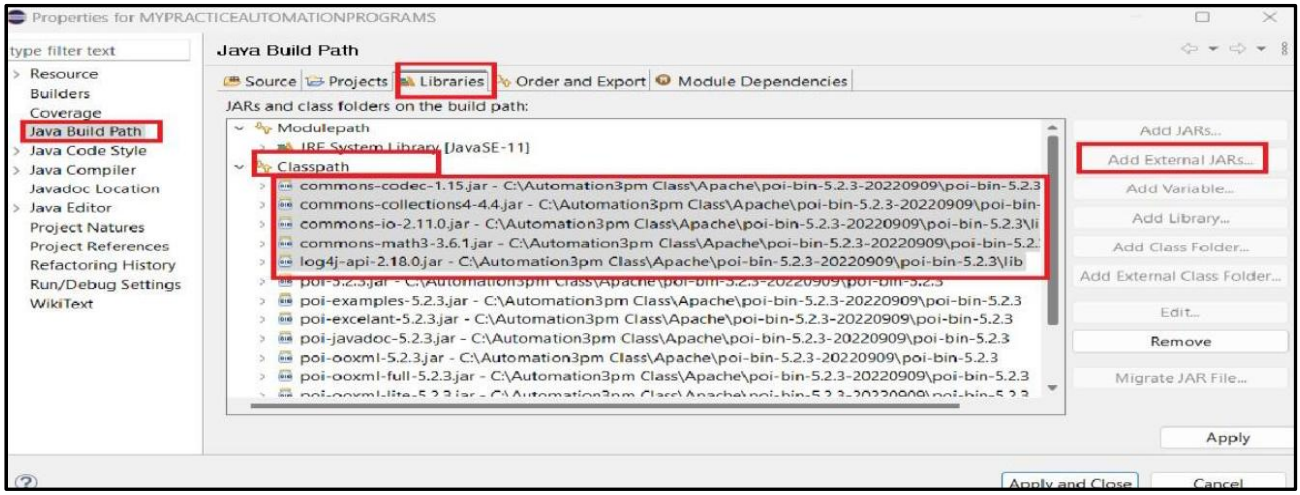
Step7: Go to the project in Eclipse, Right click on it > go to build path > configure build path:



Step8: Further click on the "add external jars" & select all the files given in the "poi-bin-5.2.3" & click on "apply and close" button:



Step9: Further click on the "add external jars" & select all the files given in the "poi-bin-5.2.3\lib" & click on "apply and close" button:



Step10: The added jar files can be found in the project under the "referenced libraries" folder.

Write Data into Excel file:

To write data to an Excel file in Selenium using Apache POI, you can follow these steps. Here's a step-by-step explanation

Step1: Create an Instance of the Workbook Class and Create a New Excel File.

```
Workbook workbook = new XSSFWorkbook();
Sheet sheet = workbook.createSheet("Sheet1");
```

In this step, you create a new instance of the Workbook class, specifically the XSSFWorkbook implementation for .xlsx files. This creates a new in-memory workbook object. Then, you create a new sheet within the workbook using the createSheet() method and provide a name for the sheet, such as Sheet1.

Step2: Create Rows and Cells and Write the Data.

```
Row row = sheet.createRow(0); cell.setCellValue("Hello, World!");
```

In this step, you create a Row object within the sheet using the createRow() method, passing the row index as the parameter. In this example, we create a row at index 0. Then, you create a Cell object within the row using the createCell() method and pass the cell index as the parameter. In this case, we create a cell at index 0 of the row. Finally, you set the cell value using the setCellValue() method. You can provide any desired value, such as Hello, World!.

Step3: Create a File Output Stream and Write the Workbook Data to a File.

```
FileOutputStream file = new FileOutputStream("path/to/excel/file.xlsx");
workbook.write(file);
```

In this step, you create a FileOutputStream object to specify the file where you want to write the data. You provide the file path as a parameter. Then, you use the write() method of the Workbook class to write the workbook data to the file specified by the FileOutputStream object.

Step4: Close the Workbook and File Output Stream.

```
workbook.close(); file.close();
```

In this step, after writing the data, you need to close the workbook and the FileOutputStream to release system resources.

Read data from Excel file:

While reading the Excel file, Apache POI can read data in two ways:

- You want to read the value of a **particular cell**, for instance, you want to get the address of the student present in the second row.
- You can read the **entire excel** in one go. It is based upon the need for your test script and the data needed for test execution.

Suppose, for a Selenium test case, we need to read the student data from the Excel Sheet, having a sample data:

Precondition:

1. Create an xlsx file and save it at particular location. Enter some data to read using Selenium. Close the created excel file before executing the script. (I have placed an excel file 'ExcelFileRead.xlsx' in my D Drive and my sheet name is "Sheet1".)
2. Go to option "Format Cells" and under Number Tab option, select Text and click on OK. By default it will be general, you need to make it as Number. If you don't make it as text, there is a chance of NullPointerException error.

To read data from excel sheet in selenium webdriver using java Apache POI, you need to follow a series of steps.

Step1: Create an Instance of the Workbook Class and Open the Excel File.

```
FileInputStream file = new FileInputStream("path/to/excel/file.xlsx");
Workbook workbook = new XSSFWorkbook(file);
```

In this step, you create a FileInputStream object to read the Excel file from the specified path. Then, you create an instance of the Workbook class, specifically the XSSFWorkbook implementation for .xlsx files, and pass the FileInputStream object as a parameter to open the workbook.

Step2: Get the Desired Sheet from the Workbook.

```
Sheet sheet = workbook.getSheet("Sheet1");
```

In this step, you use the getSheet() method of the Workbook class to retrieve the desired sheet by specifying its name. You can replace Sheet1 with the actual name of the sheet you want to read data from.

Step3: Iterate Over the Rows and Cells to Read the Data.

```
Iterator<Row> rowIterator = sheet.iterator();
while (rowIterator.hasNext()) {
Row row = rowIterator.next();
Iterator<Cell> cellIterator = row.iterator();
while (cellIterator.hasNext()) {
Cell cell = cellIterator.next();
// Process the cell value
} }
```

In this step, you create an iterator for the rows in the sheet using the iterator() method of the Sheet class. You then iterate over each row using a while loop and retrieve the Row object. Inside the row loop, you create an iterator for the cells within the row and iterate over them using another while loop. For each cell, you retrieve the Cell object.

Step4: Process the Cell Value Based on its Data Type using the DataFormatter.

```
DataFormatter dataFormatter = new DataFormatter();
String cellValue = dataFormatter.formatCellValue(cell);
```

In this step, To handle different data types of cell values, you create an instance of the DataFormatter class. The DataFormatter class helps in formatting cell values based on their data type. You use the formatCellValue() method of the DataFormatter class to get the formatted value of the cell as a String.

Step5: Close the Workbook and File Input Stream.

```
workbook.close(); file.close();
```

In this step, after reading the data, you need to close the workbook and the FileInputStream to release system resources.

****

Keyword Driven Framework

Keyword Driven Framework is a type of Functional Automation Testing Framework which is also known as Table-Driven testing or Action Word based testing. The basic working of the Keyword Driven Framework is to divide the Test Case into four different parts. First is called as Test Step, second is Object of Test Step, third is Action on Test Object and fourth is Data for Test Object.

In Selenium, a keyword driven framework is a mechanism for speeding up automated testing by separating keywords for a common set of functions and commands. All actions and instructions are written in an external file, such as an Excel sheet. Using a Keyword driven testing framework, all operations and instructions are written in an external excel file. Test automation frameworks like Selenium help separate the code and the data to help maintain test cases. This makes it easier to read, reuse, and reduce the cost of maintenance. A keyword driven framework in Selenium is a testing approach that uses a table of keywords to represent the actions and inputs for each test case. The test cases are then constructed by calling the keywords in a specific sequence to perform the desired testing steps.

Keyword-driven testing is a functional testing, also called table-driven testing or action-word based testing. In this testing, a table format is used, most likely a spreadsheet, to define keywords for a function to be executed.

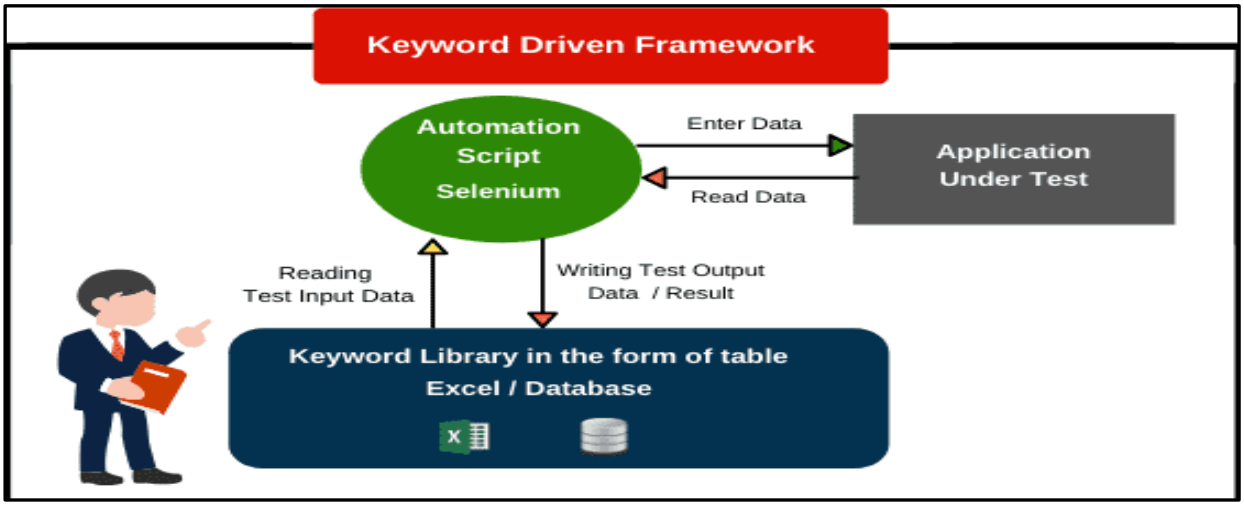
For example, consider a website. The keyword “login” is used in automation framework to test the login function associated with it. Also, the keyword “logout” is used in automation framework to test the logout function related to it.

Examples of keywords –

Keywords	Description
Login	Login to “XYZ” website
Emails	Send an email
Logouts	Logout from “XYZ” website
Notifications	Find unread notifications

Components Of Keyword Driven Testing Framework:

1. **Test Data:** External file (e.g., Excel) containing test steps, keywords, locators, and input data.
2. **Keywords:** Predefined high-level actions like openBrowser, click, type, verifyText, etc.
3. **Object Repository:** Stores element locators for easy maintenance.
4. **Driver Script:** Reads the test data, maps keywords to corresponding methods, and executes them.
5. **Utility Functions:** Reusable methods for actions like clicking a button, entering text, or verifying output.



Framework Workflow:

1. **Create Test Data:** Use an external file (Excel, CSV, or database) where each row defines a test step, including:
2. **Develop Keyword Functions:** Implement reusable functions in code (e.g., Java, Python). For example:
   - openBrowser()
   - navigate(url)
   - click(locator)
   - type(locator, value)
   - verifyText(locator, expectedValue)
3. **Build Driver Script:** This script:
   - Reads test steps from the test data file.
   - Maps each keyword to its corresponding function.
   - Executes the functions in sequence.
4. **Object Repository:** Centralized storage for locators (e.g., locators are mapped in Excel or properties file).
5. **Test Execution:** Run the framework. The driver script reads the test case and executes the corresponding actions based on the keywords and parameters.

Project Structure:

```
KeywordDrivenFramework/
|-- src/
|   |-- testcases/         // Contains test case data (Excel/CSV files)
|   |-- keywords/          // Action methods (e.g., click, inputText)
|   |-- utils/             // Utility classes (e.g., ExcelReader)
|   |-- config/            // Object Repository or config files
|   |-- driverscript/      // Main driver script
|-- lib/                   // Selenium JARs and dependencies
|-- test-output/           // Test reports
```

Framework Implementation:

Step 1: Sample Test Data (Excel):

Test Case	Keyword	Locator Type	Locator Value	Input Data
Login	openBrowser			Chrome
Login	navigate			https://example.com
Login	type	id	username	testUser
Login	type	id	password	pass123
Login	click	xpath	//button[@id='login']	
Login	verifyText	xpath	//h1[@id='welcome']	Welcome, testUser

Step 2: Dependencies (POM.xml):

```
<dependencies>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>4.11.0</version>
    </dependency>
    <dependency>
        <groupId>org.apache.poi</groupId>
        <artifactId>poi-ooxml</artifactId>
        <version>5.2.3</version>
    </dependency>
    <dependency>
        <groupId>org.testng</groupId>
        <artifactId>testng</artifactId>
        <version>7.7.1</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Step 3: Java [Script] Implementation:

→ ExcelReader Utility: Reads test data from Excel.

```
package utils;
import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import java.io.FileInputStream;
import java.io.IOException;
public class ExcelReader {
    private Workbook workbook;
    public ExcelReader(String filePath) throws IOException {
        FileInputStream fis = new FileInputStream(filePath);
```

```
        workbook = new XSSFWorkbook(fis);       }
    public String getCellData(String sheetName, int row, int col) {
        Sheet sheet = workbook.getSheet(sheetName);
        Cell cell = sheet.getRow(row).getCell(col);
        return cell.toString();       }
    public int getRowCount(String sheetName) {
        return workbook.getSheet(sheetName).getLastRowNum();       }
    public int getColumnCount(String sheetName) {
        return workbook.getSheet(sheetName).getRow(0).getLastCellNum();
    }         }
```

→ **Keyword Functions:**

```
package keywords;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
public class KeywordFunctions {
    WebDriver driver;
    public KeywordFunctions(WebDriver driver) {
        this.driver = driver;           }
    public void openURL(String url) {
        driver.get(url);           }
    public void input(String locatorType, String locatorValue, String
testData) {
        WebElement element = getElement(locatorType, locatorValue);
        element.clear();
        element.sendKeys(testData);          }
    public void click(String locatorType, String locatorValue) {
        WebElement element = getElement(locatorType, locatorValue);
        element.click();         }
    private WebElement getElement(String locatorType, String
locatorValue) {
        switch (locatorType.toLowerCase()) {
            case "id":
                return driver.findElement(By.id(locatorValue));
            case "name":
                return driver.findElement(By.name(locatorValue));
            case "xpath":
                return driver.findElement(By.xpath(locatorValue));
            case "css":
                return driver.findElement(By.cssSelector(locatorValue));
```

```
            default:
                throw new IllegalArgumentException("Invalid locator type: "
+ locatorType);
        }       }       }
```

→ **Driver Script:**

```
package driverscript;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import keywords.KeywordFunctions;
import utils.ExcelReader;
public class DriverScript {
public static void main(String[] args) {
WebDriver driver = new ChromeDriver();
KeywordFunctions kf = new KeywordFunctions(driver);
try {
ExcelReader excel = new ExcelReader("src/testcases/TestData.xlsx");
int rowCount = excel.getRowCount("Sheet1");
for (int i = 1; i <= rowCount; i++) {
String action = excel.getCellData("Sheet1", i, 1);
String locatorType = excel.getCellData("Sheet1", i, 2);
String locatorValue = excel.getCellData("Sheet1", i, 3);
String testData = excel.getCellData("Sheet1", i, 4);
switch (action.toLowerCase()) {
case "openurl":
kf.openURL(testData);
break;
case "input":
kf.input(locatorType, locatorValue, testData);
break;
case "click":
kf.click(locatorType, locatorValue);
break;
default:
System.out.println("Invalid action: " + action);
}               }
} catch (Exception e) {
e.printStackTrace();
} finally {
driver.quit();
}       }   }
```

How It Works:

- The Driver Script reads the Excel file row by row.
- It identifies the Action (e.g., openURL, click, input) and calls the corresponding method from Keyword Functions.
- The method executes the action on the application based on the Locator Type, Locator Value, and Test Data.

Advantages of the Framework:

✓ Separation of Test Logic and Data: Test steps and logic are maintained outside the code.
✓ Reusability: Keyword methods can be reused across multiple test cases.
✓ Scalability: Adding new test cases is straightforward.
✓ Non-Technical Tester Friendly: Non-programmers can define test cases in Excel.
✓ This framework provides a clean and modular approach to automation testing and is highly maintainable.

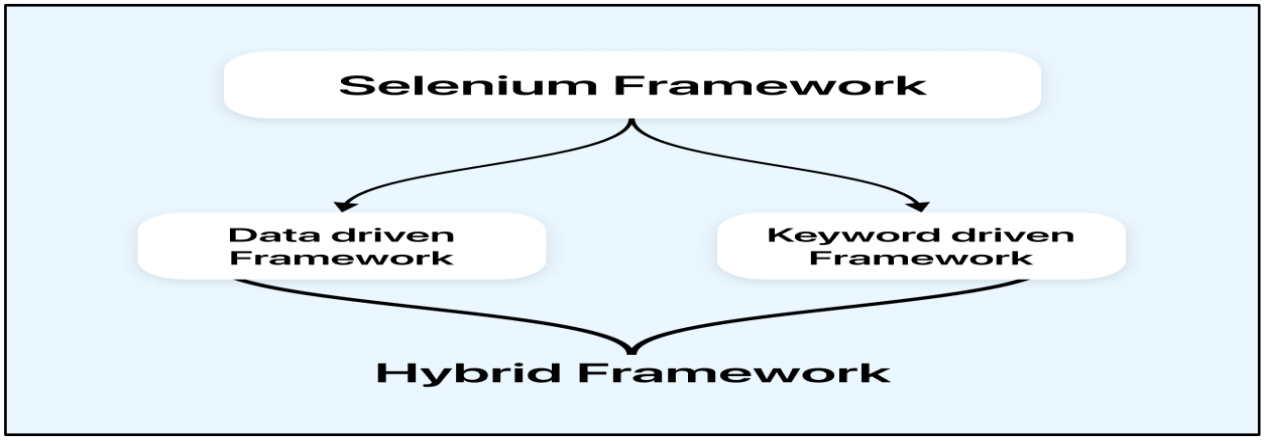
Differences: **Data Driven Testing** and **Keyword Driven Testing**

Data Driven Testing	Keyword Driven Testing
It involves testing using different stored data, values, and variables.	It involves testing using specific keywords.
Store data in various formats like CSV files, excel sheets, databases, tables, etc.	Store data in excel sheets.
The user decodes all test scripts.	The user encodes all test scripts.
It can occur in several stages.	It can occur in two different stages.
This method is less time-consuming.	This method is more time-consuming.
It is a time-consuming process.	It is the least time-consuming method.
It requires coding and technical expertise to run tests.	No coding or technical expertise is required.
It doesn't require extensive planning.	It requires extensive planning.

****

Hybrid Driven Framework

A Hybrid Framework in Selenium is a comprehensive software testing framework combining primarily two testing methodologies: Data driven and Keyword driven. Here, the keywords, as well as the test data, are externalized. Keywords are maintained in a separate Java class file and test data can be maintained either in a properties file/excel file/can use the data provider of a TestNG framework. The goal is to create a flexible and efficient testing framework that can handle various scenarios, thus enhancing test reusability, abstract technical complexities, and fostering stakeholder collaboration. The test cases created with a hybrid framework are easier to maintain and addition of new test cases to this type of framework require less of time.



Components of Hybrid Framework:

A Selenium Hybrid Framework is a strong mix of several testing approaches developed to optimize reusability, maintainability, and flexibility of test automation. Let us explore the following components to gain an extensive understanding of the Hybrid Framework.

1) Function Library **2)** Object Repository **3)** Keyword Management
4) Driver Script **5)** Test Case Template Design

1) Function Library: Function Library relates to the repository of the reusable functions or methods enclosing particular functionalities, actions, and behaviours. We can utilize these functions throughout several components or layers of the hybrid framework, like automated testing, manual testing, and other activities of software testing. For each user action, user-defined methods are created. Keywords are thus created in the library file.

2) Object Repository [Elements/Locators]: The Object Repository is the organized storage mechanism for every UI element and locator in the test automation scripts. Rather than programming the locators directly into the test scripts, we will store them in an object repository, enabling us to handle and update locators while they modify without changing test scripts. All elements on the webpage are kept in their Repository. In an Object Repository, each WebElement is identified by a name followed by its value.

3) Keyword Management [Stored in Excel Spreadsheet]: Keywords created in the library file are saved in an excel sheet with a description so anyone using this framework can

understand them. After declaring the keywords for the instances, we have to handle them. Generally, we store them in Excel sheets.

i) Keyword Definitions: Define the row in the Excel sheet for every action or the test step we have to automate. This row will include columns for several attributes of keywords, like

⇢ Action ⇢Keyword Name ⇢ Element Locator ⇢ Element Identifier
⇢ Expected Result ⇢ Parameters ⇢ Notes or Comments

ii) Parameterization and Reusability: Excel Sheet, which enables us to parameterize the keywords easily. We store the dynamic values that modify throughout the test scenarios or cases through cells.

iii) Keyword Execution Logic: We can read Excel sheets and translate the rows as keywords in the test script. For every keyword, derive the related information and run the corresponding actions using proper Selenium functions or the other testing libraries.

iv) Splitting of Concerns: Split the test logic from test data by storing the keywords in the Excel sheets. Testers can concentrate on specifying the keywords, whereas test data can be handled separately in Excel.

4) Driver Script [Test Scripts]: In the Hybrid Framework, the test script runs the test cases through the keywords specified in our function library and tracks instructions in test case templates. It connects low-level implementation details and high-level test cases. This contains the main logic for reading all the test cases from the test case template excel sheet and performing the corresponding action from the library file. The script is created using the test case template.

i) Translating Keywords: For every step in the test case, the driver script will translate the equivalent keyword and its related parameters from the test case template.

ii) Reading the Test Cases: The driver script will read test case templates, which draft the steps to be implemented for every test case.

iii) Handling Test Data: The driver script can handle recovering and managing of test data, which can be stored explicitly in the databases and spreadsheets.

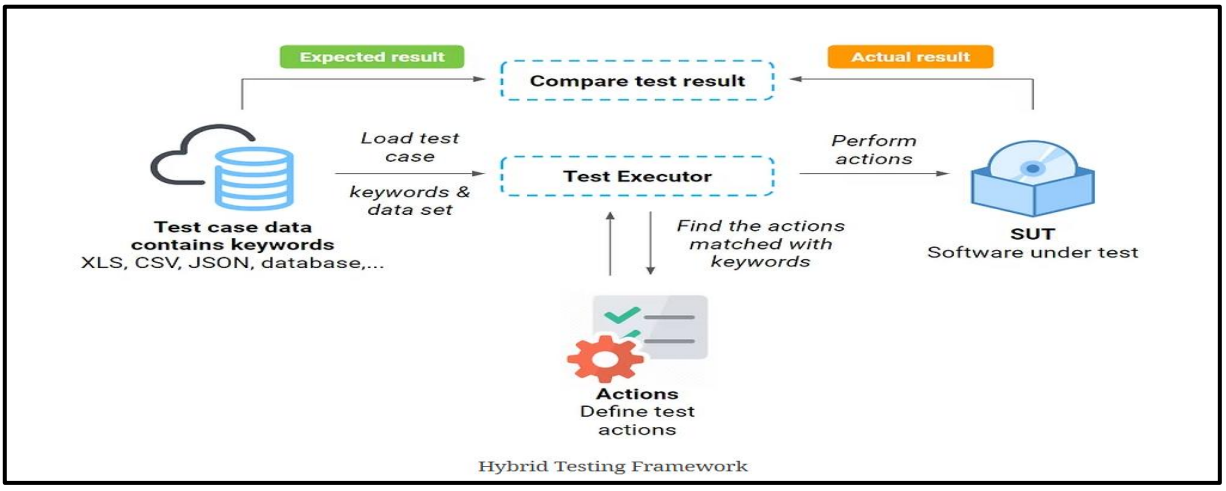
iv) Calling Functions: The driver script invokes suitable functions from the function library as per translated keywords. These functions enclose actions that have to be performed.

5) Test Case Template Design: A Consistent and well-organized test case template is needed to create consistent and organized test documentation. A well-developed template enhances the test case's thoroughness and clarity. For the framework, a Test Case template is created. There is no set pattern to follow. Both test data and keywords should be externalized, according to the Hybrid Framework.

Create a Hybrid Framework:

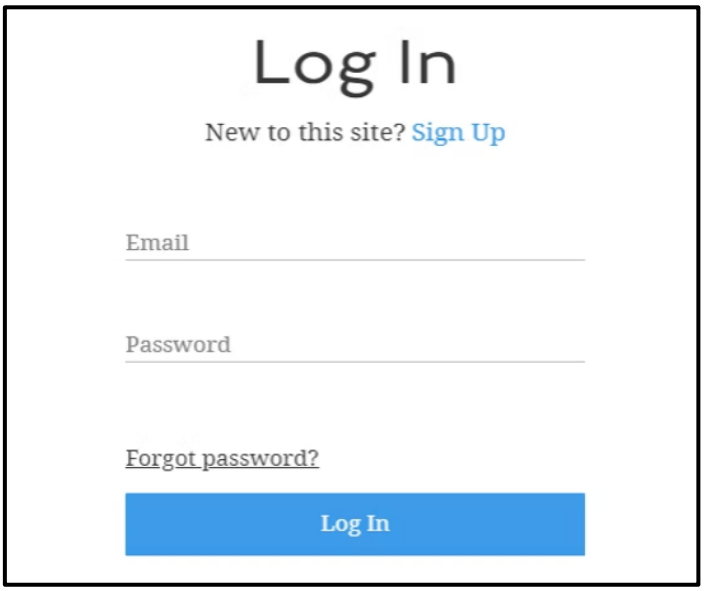
As the name suggests, the Hybrid Testing Framework is a combination of the two above mentioned frameworks. The best thing about such a setup is that it leverages the benefits of all kinds of associated frameworks.

- We use the Keyword-Driven concept to select actions to execute.
- We use the Data-Driven concept to execute a set of data within one test script.
- When we combine both, that is a Hybrid Framework.



In the example below, you will know about the concept of how to implementing a simple hybrid framework with a practical case. Consider we want to test the login functionality.

For simple, we can write multiple methods in the main test case, in which each test case will test certain functionality. For input email and password, there could be methods. And these input methods can be used multiple times with multiple data set (emails and passwords).



1. Function Library: The Function Library contains reusable functions for interacting with UI elements and performing operations. Here's an example of how you might implement some functions:

```
public class FunctionLibrary {
   WebDriver driver;
   public FunctionLibrary(WebDriver driver) {
      this.driver = driver;          }
   public void enterText(By locator, String text) {
      driver.findElement(locator).sendKeys(text);         }
```

```
    public void clickElement(By locator) {
        driver.findElement(locator).click();            }
    public String getText(By locator) {
        return driver.findElement(locator).getText();          }        }
```

2. Object Repository: The Object Repository stores UI element locators centrally for easy maintenance. You can use a **Properties** file or an external file like Excel to store these locators.

```
emailField=//input[@type='email']
passwordField=//input[@type='password']
loginButton=//button[text()='Log In']
```

Code to load locators:

```
public class ObjectRepository {
    Properties properties;
    public ObjectRepository(String filePath) throws IOException {
        properties = new Properties();
        FileInputStream fis = new FileInputStream(filePath);
        properties.load(fis);     }
    public By getLocator(String key) {
        String locator = properties.getProperty(key);
        return By.xpath(locator);
    }   }
```

3. Keyword Management: Keywords represent actions like "EnterText", "ClickButton", etc. A mapping between keywords and actions ensures modularity.

Example of Keyword Execution:

```
public class KeywordExecutor {
    FunctionLibrary functionLibrary;
    ObjectRepository objectRepository;
    public KeywordExecutor(WebDriver driver, String objectRepoPath) throws
IOException {
        functionLibrary = new FunctionLibrary(driver);
        objectRepository = new ObjectRepository(objectRepoPath);      }
    public void executeKeyword(String keyword, String objectKey, String data) {
        By locator = objectRepository.getLocator(objectKey);
        switch (keyword) {
            case "EnterText":
                functionLibrary.enterText(locator, data);
                break;
            case "ClickButton":
                functionLibrary.clickElement(locator);
```

```
            break;
        default:
            throw new IllegalArgumentException("Invalid keyword: " + keyword);
    }   }   }
```

4. Driver Script: The Driver Script drives the entire framework. It reads test cases and executes the steps using the Function Library and Keyword Management.

Example:

```
public class DriverScript {
   public static void main(String[] args) throws IOException {
      WebDriver driver = new ChromeDriver();
      driver.get("https://example.com/login");
      KeywordExecutor keywordExecutor = new KeywordExecutor(driver,
"path/to/objectRepository.properties");
      // Simulate reading test cases from an Excel file or other source
      keywordExecutor.executeKeyword("EnterText", "emailField",
"user@example.com");
      keywordExecutor.executeKeyword("EnterText", "passwordField",
"password123");
      keywordExecutor.executeKeyword("ClickButton", "loginButton");
      driver.quit();
   }  }
```

5. Test Case Template Design: Test case templates should define the flow of execution with keywords and test data. Use an Excel sheet or similar format to maintain this data.

Excel Test Case Template Example:

TestCaseID	Keyword	ObjectKey	TestData
TC_01	EnterText	emailField	user@example.com
TC_01	EnterText	passwordField	password123
TC_01	ClickButton	loginButton	

How It Works:

Initialization:

- Load the Object Repository (.properties file).
- Read the test steps from the Excel file.

Execution:

- For each test step:
    - Fetch the keyword, objectKey, and testData.
    - Call the executeKeyword method of KeywordExecutor.

Validation & Reporting:

- Print or log errors for failed steps.

- Report success for completed steps.

Advantages of Hybrid Framework:

1) Easy Testing: Software Testers can concentrate on building concise and clear test scripts, whereas the framework manages the difficulties of test reporting and execution, eventually enhancing the effectiveness of the testing reports. Testers can create high-level keywords that outline complex interactions or actions in the application.

2) Resilience to Blend Methods: The Hybrid Framework allows the software testers to continuously blend the keyword-driven and Data-Driven Frameworks to fit the project's requirements.

3) Components Reusability: Through the Data Driven Framework, testers can separate test data from test scripts and create reusable test scripts that can be utilized for different test scenarios. The component reusability lowers the efforts needed for creating new tests and assures continuous testing throughout different modules, resulting in higher test coverage.

4) Best Team Collaboration and Communication: Systematic procedures and the shared testing strategy allow a cohesive framework. Keyword-driven testing improves the test case clarity for no-technical stakeholders in the hybrid framework. It eases test design language, making it usable for contributions and reviews by non-technical associates.

5) Modular Checks: The modular approach improves maintainability since modifications in one module will not affect other modules, providing rapid updates and troubleshooting. Keyword-driven testing comes with this feature.

6) Simple Maintenance: When changes or updates are needed, testers can focus on particular modules, diminishing the effect on the other elements of the testing suite and decreasing the whole maintenance workload.

7) Wider Scenario Coverage: This assures that different aspects of the application, like functionality, performance, and user experience, are carefully tested, leading to a stronger and more trustworthy software product.

****

BDD - Cucumber with Java - Selenium Framework

BDD, or Behavior-Driven Development, is a collaborative approach for software teams that bridges the communication between business and technical professionals. The Cucumber BDD framework is a popular open-source tool that can be used with Selenium and Java for behavior-driven development (BDD) automation testing of web applications. The tests are first written in a simple scenario form that describes the expected behavior of the system from the user's perspective. Largely used for acceptance tests, Cucumber is written in Ruby, while the tests are written in Gherkin, a non-technical and human-readable language. Automation Testing accelerates the software development lifecycle. Furthermore, on adopting the Behavior Driven Development (BDD) approach to development, different stakeholders like developers, QAs, and non-tech teams can collaborate actively in the project. Widely beneficial for User Acceptance Testing where the test scenarios are largely driven by behavior, Cucumber strengthens Automation Testing. Most of the organizations use Selenium for functional testing. These organizations which are using Selenium, want to integrate Selenium with Cucumber as Cucumber makes it easy to read and to understand the application flow.

Example:

If we are developing a user authentication feature, then the following can be few key test scenarios, which needs to get passed in order to call it a success.

- The user should be able to login with correct username and correct password.
- The user should not be able to login with incorrect username and correct password.
- The user should not be able to login with correct username and incorrect password.
- The user should not be able to login with incorrect username and incorrect password.

It achieves this by:

- Fostering collaboration across different roles to cultivate a collective comprehension of the issue.
- Embracing swift, incremental iterations to enhance feedback loops and value delivery.
- Generating system documentation that undergoes automatic validation against the system's actual behavior.

Features:

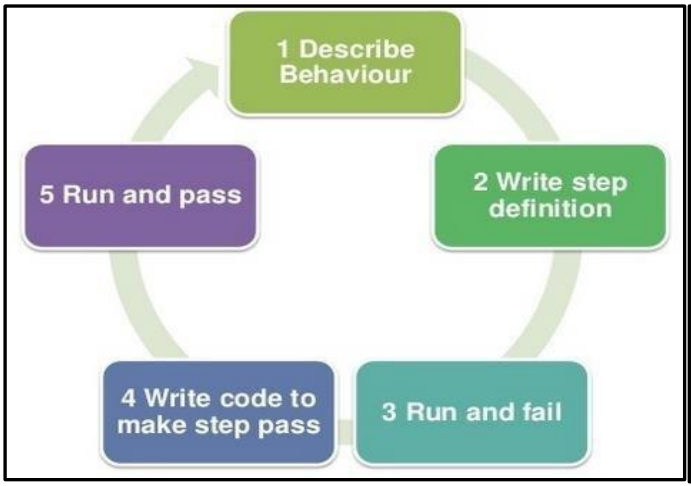
The Cucumber BDD framework has three main parts:

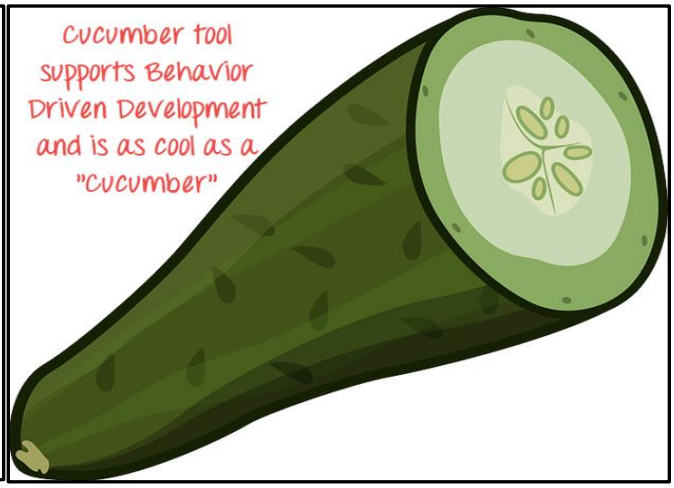
Step Definitions, Test Runner File, and Cucumber Hooks.

- Step Definitions: Maps each step of a scenario to the code that will be executed
- Test Runner File: Contains the location of the Step Definitions and other metadata for running the test
- Cucumber Hooks: Blocks of code that run before or after scenarios or steps to prepare or clean up test environments, initialize variables, and more.

The scenarios are written based on the expected behavior of the software and it is tested to check if it matches said scenarios. These scenarios are documented using a Domain Specific Language such as Gherkin. In each test scenario, natural language constructs constituting

small English-like phrases are used to describe the behavior and expected outcome of an application. This is done using a dedicated software tool like Cucumber, that allows the execution of automated acceptance tests written in Gherkin.





Advantages [Benefits] of Cucumber over Other Tools:

- Cucumber supports different languages like Java.net and Ruby.
- It acts as a bridge between the business and technical language. We can accomplish this by creating a test case in plain English text.
- It allows the test script to be written without knowledge of any code, it allows the involvement of non-programmers as well.
- It serves the purpose of end-to-end test framework unlike other tools.
- Due to simple test script architecture, Cucumber provides code reusability.
- It is helpful to involve business stakeholders who can't easily read code
- Cucumber Testing tool focuses on end-user experience
- Style of writing tests allow for easier reuse of code in the tests
- Quick and easy set up and execution
- Compatible with Agile development processes.
- Easily integrates with various CI tools and programming languages.
- Supports automated testing for CI/CD pipelines.
- Reduces misunderstandings and defects due to clear syntax.
- Cucumber test tool is an efficient tool for testing

Cucumber vs Selenium vs QTP

Cucumber	Selenium	QTP
Cucumber software is free	It is free	QTP is expensive
Cucumber software is a behavior driven development tool	It's a Functional and Performance (Selenium Grid) test tool	It's a Functional Automation Tool
Plugin in Cucumber testing tool works faster	Plugins are slower than cucumber	Plugin are slower compare to Cucumber and Selenium

Cucumber Framework supports other language as well beyond Ruby like Java, Scala, Groovy etc.	Selenium supports Java, .Net and many other languages	QTP supports only VB script
Writing automation steps are joint effort of testers and developer	Like Cucumber Tool, writing automation steps are joint effort of testers and developer	In QTP only tester writes automation steps
Cucumber testing tool supports only web environment	Supports only web environment	Support web, desktop and any client server application

Setting up Cucumber BDD Framework for Selenium:

Before exploring how Cucumber works and how to write a Cucumber Selenium test, let's first set up Cucumber and Install the prerequisites.

Prerequisites for Cucumber and Selenium Setup

- **Install/ Configure Selenium-Java** [Java is a robust programming language. Cucumber supports Java platform for the execution]

Add Selenium Java Dependencies in Eclipse Maven Project: >Open Eclipse IDE. >Create/Open Maven project. >Add Selenium Java Dependency in pom.xml. >Add the Selenium Java dependency inside the <dependencies> tag.

- **Install/Configure Eclipse IDE for Java** [Eclipse is an Integrated Development Environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment]

Install Eclipse IDE: Go to the official Eclipse download page: Eclipse Downloads>Click on the Download button for the version you want [Eclipse IDE for Java Developers]>After downloading the Eclipse (.exe file), double-click on it to launch>Click Install>Accept the license agreement and wait for the installation to complete>Once installed, you can launch Eclipse directly from the installer.

- **Install/ Configure Maven** [Maven is a build automation tool used primarily for Java projects. It provides a common platform to perform activities like generating source code, compiling code, packaging code to a jar, etc. Later if any of the software versions gets changed, Maven provides an easy way to modify the test project accordingly]

Create a Maven Project in Eclipse: >Open Eclipse >Go to File → New → Project >Select Maven Project and click Next >Provide the Group Id and Artifact Id > Click Finish.

[Group Id: It is like a package name (e.g., com.mycompany.app). Artifact Id: The name of your project (e.g., SeleniumTest).]

- **Install/ Configure Selenium Webdriver** [Selenium is an automation tool for Functional Testing of the web-based application. Selenium supports different language like java, ruby, python C#, etc.Most of the organizations use Selenium for functional testing. These

organizations which are using Selenium, want to integrate Selenium with Cucumber as Cucumber makes it easy to read and to understand the application flow.]

To add Selenium Webdriver in an Eclipse Maven project, follow these steps: >Open Eclipse IDE. >Create a new Maven project. >Add **Selenium Webdriver** Dependency in pom.xml. >Add the **Selenium Webdriver** dependency inside the <dependencies> tag.

- **Install/ Configure Junit:** JUnit is an open-source unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks.

Benefits: 1. Unit has a Graphical User Interface (GUI), making it possible to write and test source code quickly and easily. **2.** JUnit allows the developer to incrementally build test suites to measure progress and detect unintended side effects. **3.** Test can be run continuously. **4.** JUnit shows test progress in a bar that is normally green but turns red when a test fails.

On one hand, Cucumber is providing a way for non-technical person to define test cases for a product, and on the other hand, our expectation is for smooth and timely execution of such test cases.

JUnit acts as a bridge between these two. So, the flow of execution will look like the following:

- Stakeholders write down the feature file.
- Step definition file will be created accordingly.
- Specify the JUnit runner class to run the series of test cases.
- Once we run the JUnit runner class −
- It will parse the Gherkin feature file.
- It will execute the functions written in the step definition file according to feature file statements.
- JUnit will combine the test case result.
- It will build the test report in the specified format (which can be html/JSON).

To add JUnit in an Eclipse Maven project, follow these steps: >Open Eclipse IDE. >Create a new Maven project. >Add JUnit Dependency in pom.xml. >Add the JUnit dependency inside the <dependencies> tag.

- **Install/ Configure Cucumber:** Cucumber is a testing tool that supports Behavior Driven Development (BDD) framework. It defines application behavior using simple English text, defined by a language called Gherkin. Cucumber allows automation functional validation that is easily read and understood. Cucumber was initially implemented in Ruby and then extended to Java framework. Both the tools support native JUnit.

To install Cucumber in Eclipse, you can do the following: >Launch the Eclipse IDE then go to Help menu, and click "Install New Software" on this window, click the "Add" button. >After clicking the "Add" button, give the Name in the in the text box as per your choice. We provided "Cucumber". >Now, in the Location text box type " http://cucumber.github.com/cucumber-eclipse/update-site " as location then click OK.

>You will see "Cucumber Eclipse Plugin" in the software list. Just click "Check Box" and then the "Next" button. >Click the check box "I accept the terms of the license agreement" on the license window then click Finish. Now, the installation will be started. It can take some time to be completed. >The installation has been completed, now just click "Yes" button.

Add Cucumber to a Maven Project in Eclipse: >Create a Maven Project in Eclipse. >Add Cucumber Dependencies to pom.xml. >Add the Cucumber dependency inside the <dependencies> tag.

All Sample Dependencies for Cucumber Project:

Selenium-Java:

```
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
<dependency>
 <groupId>org.seleniumhq.selenium</groupId>
 <artifactId>selenium-java</artifactId>
 <version>4.25.0</version>
</dependency>
```

Junit:

```
<!-- https://mvnrepository.com/artifact/junit/junit -->
<dependency>
 <groupId>junit</groupId>
 <artifactId>junit</artifactId>
 <version>4.13.2</version>
 <scope>test</scope>
</dependency>
```

Cucumber-Java:

```
<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->
<dependency>
 <groupId>io.cucumber</groupId>
 <artifactId>cucumber-java</artifactId>
 <version>7.20.1</version>
</dependency>
```

Cucumber-Junit:

```
<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-junit -->
<dependency>
 <groupId>io.cucumber</groupId>
 <artifactId>cucumber-junit</artifactId>
 <version>7.20.1</version>
 <scope>test</scope>
</dependency>
```

Cucumber-PicoContainer:

```
<!--https://mvnrepository.com/artifact/io.cucumber/cucumber-picocontainer -->
<dependency>
 <groupId>io.cucumber</groupId>
 <artifactId>cucumber-picocontainer</artifactId>
 <version>7.20.1</version>
</dependency>
```

Three principles of BDD are:

→ Focus on the desired behavior or outcomes
→ Collaboration between developers, testers, and business stakeholders
→ Use of a common language for communication and understanding.

Gherkin Language:

Gherkin is a structured language that is used to write BDD tests. It uses a specific syntax that is easy to read and understand for both technical and non-technical team members. Gherkin scenarios are written in a Given-When-Then format to describe the expected behavior of the system in a clear and concise manner. Mastering Gherkin is essential for effective BDD development, as it ensures clarity and precision in defining user stories.

Gherkin is a business readable language which helps you to describe business behavior without going into details of implementation.

Cucumber testing framework primarily utilizes Gherkin, a domain-specific language designed for writing test cases in a way that is understandable by all stakeholders. Gherkin's syntax is straightforward and human-readable, making it an ideal choice for expressing software behavior without delving into technical details.

Gherkin - A Universal Language for Testing:

Gherkin's simplicity is its strength. It uses plain English (or other supported languages) with keywords like feature, scenario, given, when, then, and but. This approach allows for the creation of clear, concise, and readable test cases.

Support for Multiple Programming Languages:

While Gherkin is used for writing test scenarios, Cucumber seamlessly integrates these scenarios with step definitions written in various programming languages. This includes widely used languages like Java, Ruby, and JavaScript, allowing teams to leverage their existing programming skills and resources.

The Role of Gherkin in Cucumber Testing:

Gherkin acts as a bridge between technical and non-technical teams, helping developers to business analysts collaborate effectively. This universality makes Gherkin and by extension Cucumber, a powerful tool in modern software development.

The need for Gherkin can be easily explained by following images:

Before Gherkin **After Gherkin**

In BDD, "Given-When-And-Then-But" is the proposed approach for writing test cases. Listed below is an example of BDD.

```
test > features > featureFiles > platform > test.feature > ...
  1   Feature: User wants to login into the site with valid data and Invalid data
  2
  3     Background:
  4       Given User navigates to the Website
  5
  6     Scenario: Login as a new sign-up user with valid data
  7       When User enters the user name
  8       And User enters the password
  9       And User clicks on the sign-in button
 10       Then User should be logged-in successfully
 11       But Log-in button is not present
 12
 13     Scenario: Login as a new sign-up user with invalid data
 14       When User enters an invalid user name
 15       And User enters the password
 16       And User clicks on the sign-in button
 17       Then Error message should be displayed
 18       But Re-login option should be available
 19
 20
```

In the above example, there are many keywords, and every keyword has its own meaning and purpose.

- **Feature Keyword:** The feature file starts with the keyword Feature. Under feature, you can mention the feature name which is to be tested, as seen below,
  Feature: User login into the site with valid data and Invalid data
- **Scenario Keyword:** There can be more than one scenario under one feature file, and each scenario starts with the keyword Scenario, followed by the scenario name. In the below example, we can see there are two scenarios.
  Scenario: Login as a new sign-up user with valid data
  Scenario: Login as a new sign-up user with invalid data
- **Given Keyword:** Given keyword is normally used when we have to give some pre-condition in our test case as seen in the example below,
  Given User is on the home page of the site Or Given User is on the Log-in page of the site

- **When Keyword:** When the keyword is used to perform some action, as depicted in the example below,
  When the User enters an email Or When the User clicks on the "Log in" button
- **And Keyword:** There is no annotation for AND keyword. It is used to connect two statements and provides the logical AND condition between any two statements. AND can be used in combination with GIVEN, WHEN and THEN statements and used for positive assertion.
  And the user clicks on the login button
- **Then Keyword:** Then Keyword is used for the final outcome or for validation.
  Then User should be logged-in successfully
  Then Error message should be displayed
- **But Keyword:** But the keyword is used to represent negative assertions in addition to the previous statement.
  But Log-in button is not present
  But Re-login option should be available
- **Background Keyword:** If there is any repetitive step that is going to be used in every test case. We just add the repetitive step under Keyword Background. Step keep under Background would be run before every test case.

Background: Given User navigates to the Website

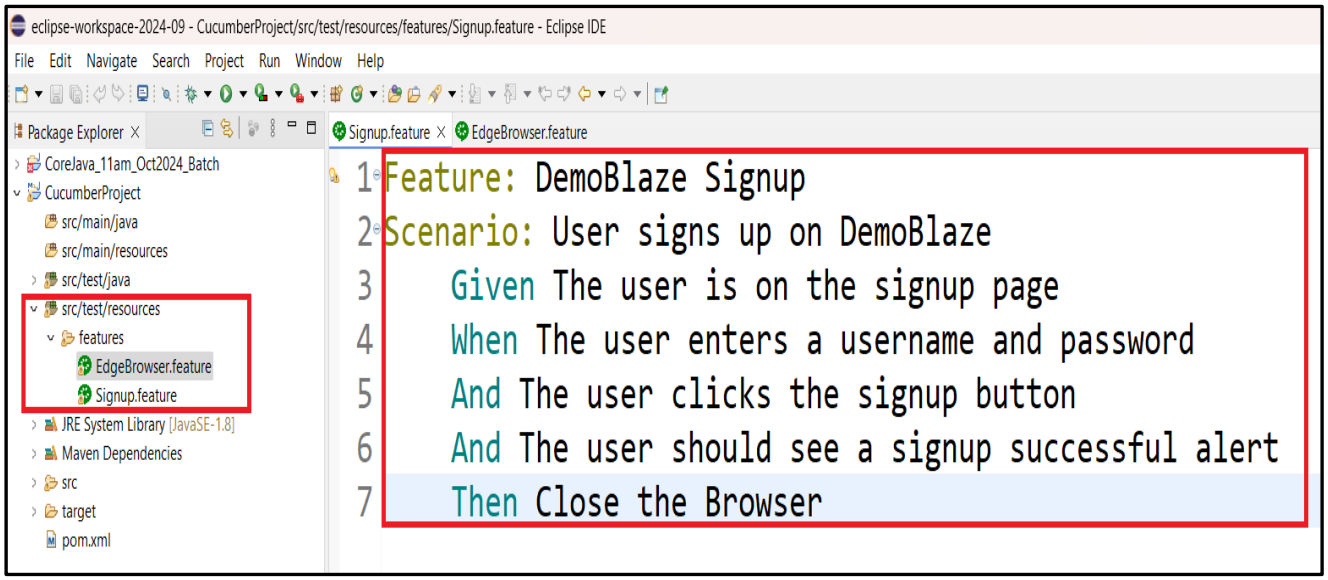
How to Start/ Set Up with Cucumber Testing: Setting up Cucumber testing framework involves several key steps to ensure a smooth and efficient testing process.

Choose a Programming Language: Decide on the programming language you will use for writing your step definitions. Cucumber supports several languages, including Java, Ruby, and .NET.
Install Cucumber: Install Cucumber on your system. This process will vary depending on your chosen programming language and development environment. ▪ Open Eclipse and go to Help > Eclipse Marketplace. ▪ In the search bar, type Cucumber and press Enter. ▪ Look for Cucumber Eclipse Plugin in the results. ▪ Click Install and follow the on-screen instructions to complete the installation. ▪ Once installed, restart Eclipse. After restarting, you should Verify the Installation: Window > Preferences > Cucumber category.
Choose Your Development Environment: Select an Integrated Development Environment (IDE) that supports Cucumber and your preferred programming language (Eclipse).
Set Up Your Project: Create a new project in your IDE and set up the necessary directories for your feature files and step definitions.

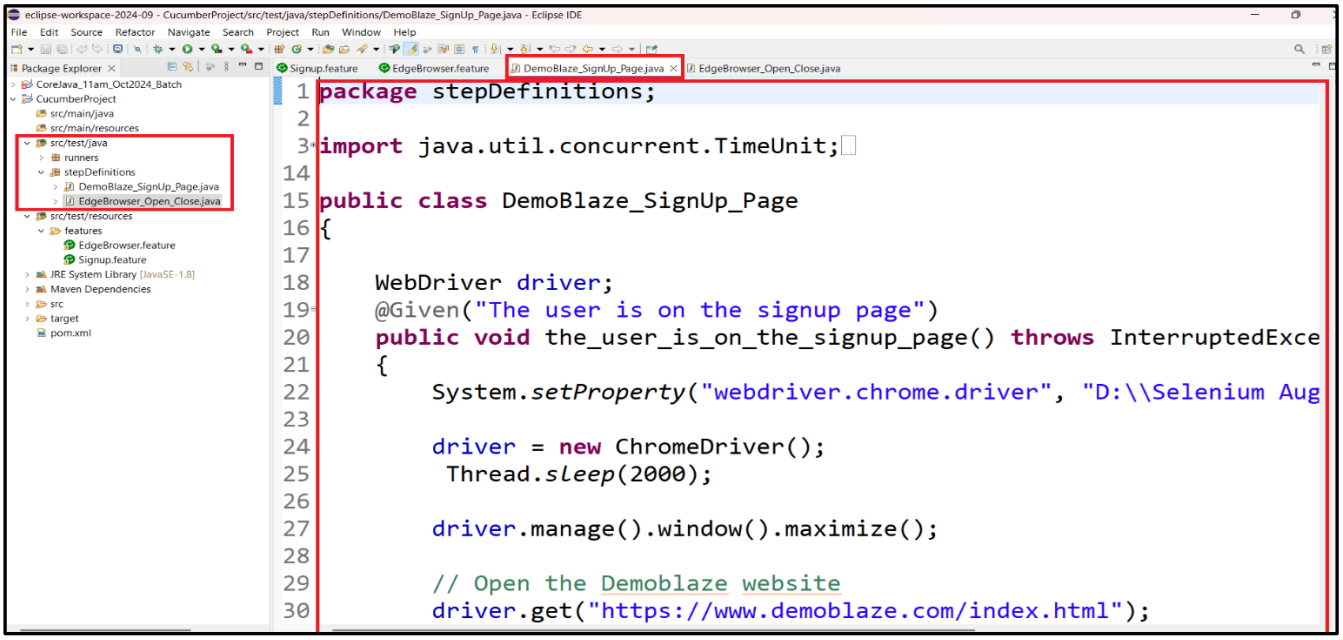
Write Your [Set Up] First Feature File: Create a feature file in Gherkin. This file will contain your first test scenario, described in a format that Cucumber can understand. In the 'features' directory, create a .feature file and write your first scenario using Gherkin syntax.
Write/ Implement Step Definitions: For each step in your Gherkin scenario, write corresponding step definitions. These are the actual code snippets that will execute the steps of your test. Create a new directory for your step definitions. Write the Java code that will execute the steps outlined in your feature file.
Configure Cucumber Runner: Set up a Cucumber runner class. This class will be used to execute the feature files and step definitions.
Run Your Test: Execute the test to see if your scenario passes or fails. This will validate whether the software behaves as expected according to your Gherkin scenario.

How does Cucumber Work: Cucumber BDD framework mainly consists of three major parts:

Feature File: Cucumber tests are written in plain text files called feature files stored with the extension – ".feature". A Feature File can be described to make the documentation more legible. These files describe the behavior and functionality of the software using a specific syntax called Gherkin. Gherkin is a structured language that uses keywords like Given, When, and Then to define the steps of a test scenario.

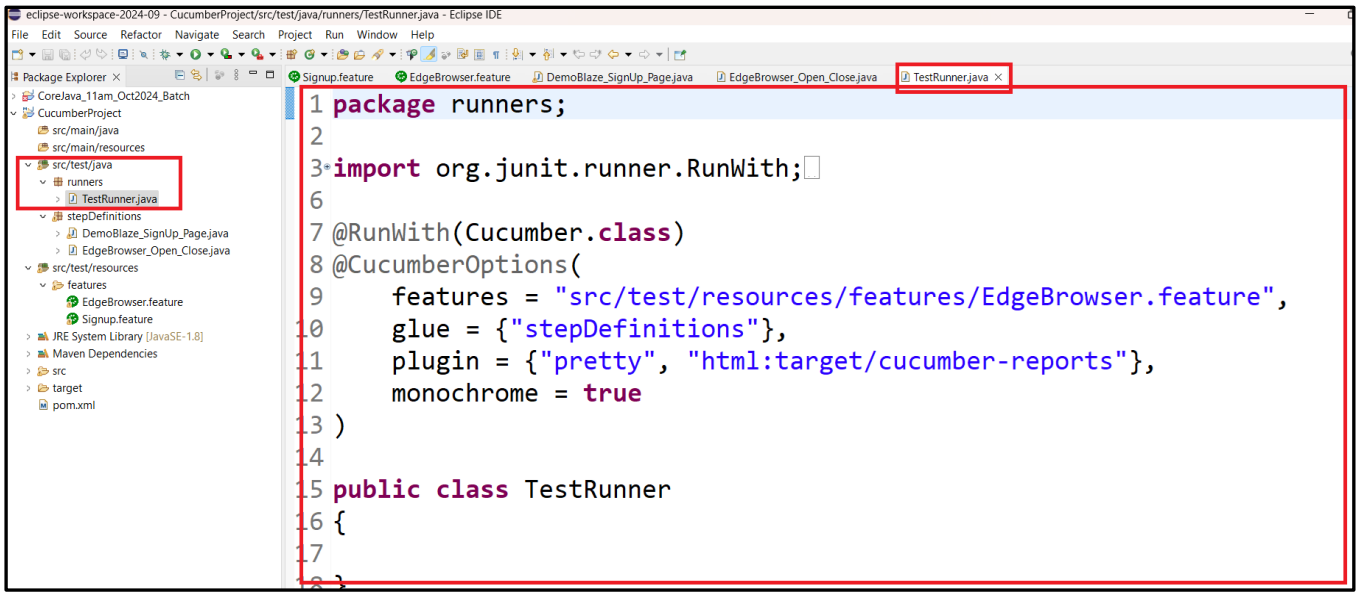


Step Definitions: Each step in a feature file is associated with a step definition implemented in the code. Step definitions define the actions or operations that must be executed for each step of the test scenario. They map the plain text steps in the feature file to the corresponding code implementation.



```
package stepDefinitions;

import java.util.concurrent.TimeUnit;

public class DemoBlaze_SignUp_Page
{

    WebDriver driver;
    @Given("The user is on the signup page")
    public void the_user_is_on_the_signup_page() throws InterruptedExce
    {
        System.setProperty("webdriver.chrome.driver", "D:\\Selenium Aug

        driver = new ChromeDriver();
         Thread.sleep(2000);

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

        // Open the Demoblaze website
        driver.get("https://www.demoblaze.com/index.html");
```

Test Runner File: In Cucumber, the test runner file executes the Cucumber feature files and coordinates the steps defined in those feature files with the corresponding step definitions.



```
package runners;

import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(
    features = "src/test/resources/features/EdgeBrowser.feature",
    glue = {"stepDefinitions"},
    plugin = {"pretty", "html:target/cucumber-reports"},
    monochrome = true
)

public class TestRunner
{

}
```

Explanation about Test Runner File:

@RunWith annotation is used to specify a custom test runner for running JUnit tests. By default, JUnit uses its own test runner, but you can use the @RunWith annotation to use a different one. In this case, Cucumber.class is provided to use Cucumber's test runner instead of the default JUnit runner.

Cucumber.class: Cucumber provides a test runner class Cucumber.class, which tells JUnit to run the tests as Cucumber tests. It looks for feature files (written in Gherkin) and step definitions (the Java methods that execute the Gherkin steps).

@CucumberOptions annotation in Cucumber is used to define various options for running your Cucumber tests. It provides a way to customize the behavior of the test execution, such as specifying the location of feature files, plugins for reporting, and more. This annotation is typically placed above the test runner class in Cucumber-based frameworks (like in a

JUnit or TestNG runner class). Here is an explanation of the most common options available within @CucumberOptions.

1. features: This option is used to specify the path to the feature files. You can point to the folder where the .feature files are located.

@CucumberOptions(features = "src/test/resources/features")

2. glue: This specifies the package where step definition methods are located (i.e., the location of your Java classes that implement the steps).

@CucumberOptions(glue = "com.example.stepdefinitions")

3. tags: This option allows you to run tests based on tags assigned to scenarios in the feature files. You can include or exclude certain tests based on their tags.

@CucumberOptions(tags = "@SmokeTest")

4. plugin: Used to specify different types of reporting options (such as pretty output, HTML reports, JSON, etc.).

@CucumberOptions(plugin = {"pretty", "html:target/cucumber-reports"})

5. monochrome: This option makes the console output more readable by removing unnecessary characters (like ANSI color codes). It is usually set to true.

@CucumberOptions(monochrome = true)

6. dryRun: When dryRun is set to true, Cucumber checks if every step in the feature files has corresponding step definitions without actually running the test code.

@CucumberOptions(dryRun = true)

7. strict (Deprecated in newer versions): This option was used to fail the test execution if there were undefined or pending steps. It's deprecated now and handled differently in recent Cucumber versions.

@CucumberOptions(strict = true)

8. name: Used to filter scenarios by their names. You can specify part or the full name of a scenario to be executed.

@CucumberOptions(name = "Login test")

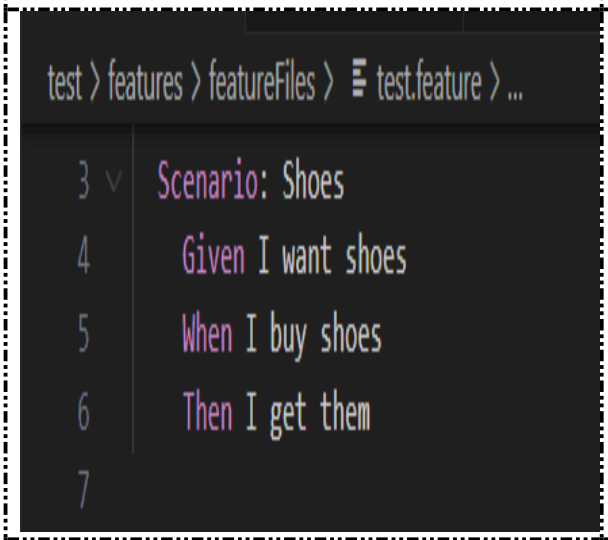
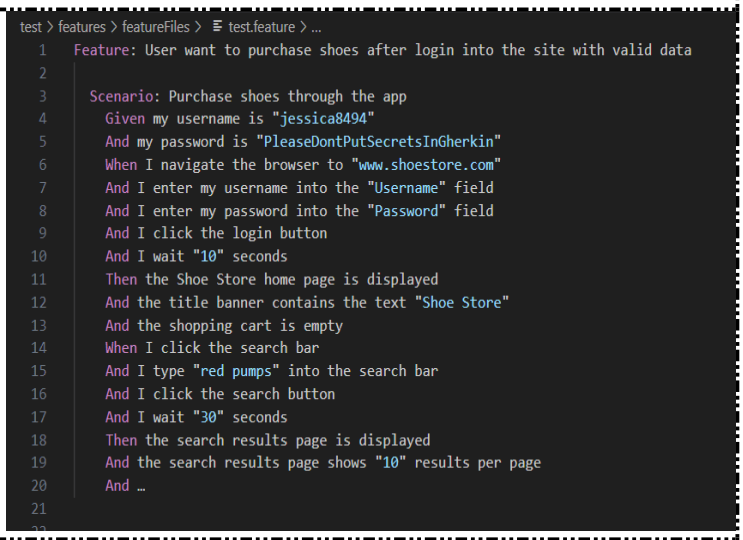
Example:	**In this example:**
@RunWith(Cucumber.class) @CucumberOptions(features = "src/test/resources/features", glue = "com.example.stepdefinitions", plugin = {"pretty", "html:target/cucumber-reports"}, monochrome = true, tags = "@SmokeTest", dryRun = false)	features points to where the feature files are located. glue points to the package with the step definitions. plugin generates readable output and HTML reports. monochrome makes the console output clean. tags filters the tests to only run scenarios tagged with @SmokeTest. dryRun is set to false to execute the tests normally (not just checking for step definitions).

public class TestRunner {
}
 	
	This setup helps in controlling the behavior of your Cucumber tests at runtime.

Gherkin's Golden Rule is Simple:

Treat other readers as you would want to be treated. Write feature files in such a way that they are easily understandable by everyone involved. Good Gherkin enhances team collaboration by clarifying the behaviors being developed. When Gherkin is difficult for others to comprehend, it hinders the development of good behaviors.

To illustrate this rule, let's examine scenarios from the perspective of a team developing a website for a shoe store:

Ultra-Declarative Scenario	Ultra-Imperative Scenario
	
This scenario lacks specificity and clarity. It doesn't provide enough details about the desired behavior, making it difficult to understand what actions are being performed and how to verify them. It's not automatable and lacks accountability.	This scenario provides excessive detail, overwhelming the reader with a lengthy list of low-level interactions. While each step is automatable, collaboration becomes challenging due to the verbosity and lack of focus.

Optimized Scenario:

```
test > features > featureFiles > test.feature > ...
1   Feature: User want to purchase shoes
2
3     Scenario: Add shoes to the shopping cart
4       Given the shoe store home page is displayed
5       When the shopper searches for "red pumps"
6       And the shopper adds the first result to the cart
7       Then the cart has one pair of "red pumps"
8
```

This scenario follows the given-when-then structure, providing clear steps that describe the setup, interaction, and verification of the behavior. It uses concrete examples like “red pumps” to enhance understanding. The scenario is concise, unambiguous, and facilitates collaboration.

What is Gherkin Syntax?

"Gherkin uses a set of special keywords to give structure and meaning to executable specifications. In the Gherkin language, lines mostly start with these keywords."

The primary/Widely used keywords in Gherkin are listed below:

Feature, Scenario, Given, When, Then, And, But, Example.

Feature - In the feature, we can describe the module, like on which module we are working or creating the feature file.

Scenario - In the Scenario, we are describing our TC or scenario.

Given - The given keyword describes the initial context (Prerequisites).

When - When a keyword is used to describe the Action.

Then - In Then, we add the Expected result or what we expect.

And, But: Used to combine multiple Given, When, or Then steps.

Example -Provides a table of inputs to a Scenario Outline

Different Types of Cucumber Testing: Cucumber's adaptability allows it to be used for various testing types. Each type addresses specific aspects of software quality assurance:

Functional Testing:	Verifying that each software function operates in conformance with the required specification. This involves creating feature files that describe and test various functionalities to ensure they meet business requirements.
Regression Testing:	Focused on identifying regressions, this involves re-running tests that have previously passed to ensure that recent changes haven't adversely affected existing functionality.
End-to-End Testing:	Assessing the complete flow of an application from start to finish, ensuring all integrated components work together as expected. This

	typically covers multiple features and scenarios, simulating real user scenarios.
Unit Testing:	These tests help you determine if a specific unit of code is working as expected. Unit tests are typically written in a programming language like Ruby or JavaScript.
Integration Testing:	Concentrating on the points where different system components interact, this type of testing ensures that modules or services work together seamlessly, highlighting interface defects.
Acceptance Testing:	Aimed at verifying the system fulfills business requirements, often based on user stories. Acceptance tests confirm that the overall functionality is as expected by the end user.
Smoke Testing:	A preliminary testing type to check the basic functionality of the application. It's a quick, high-level test to ensure the major features of the software are working correctly.

You can enhance your Cucumber tests by adding features like data tables, screenshots, and more. These features can help you make your tests more robust and easier to understand.

Data Tables: Data tables can be used to provide input data for your Cucumber tests. Data tables can be defined in your .feature files or can be external files. You can use data tables to test different input values or run the same scenario with varying data sets.

For example, you could use a data table to test a login feature with a different username and password combinations. To use a data table, you must define the header row and the data rows in your .feature file.
The header row defines the variable names used in the data rows. The data rows contain the actual data used in the test.

```
Scenario Outline: Login feature
Given I am on the login page
When I enter my username and password
Then I should be logged in

Examples:
| username | password |
| user1 | pass1 |
| user2 | pass2 |
| user3 | pass3 |
```

ScreenShots: Screenshots can be taken automatically when a Cucumber test fails. These screenshots can help you debug your tests and understand why they might have failed. You can configure Cucumber to take screenshots automatically by setting the 'screenshot' option in your Cucumber.yml file.

In this example, the take_screenshot method takes a screenshot and embeds it in the HTML report. The method takes a screenshot of the login page and saves it as a png file. You can view the screenshot in the HTML report that Cucumber generates.	``` Scenario: Login feature Given I am on the login page When I enter my username and password Then I should be logged in And I take a screenshot def take_screenshot embed("screenshot.png", "image/png") end ```

HTML Reports: HTML reports can be generated to show the results of your Cucumber tests. These reports can be used to share the results of your tests with others. You can generate HTML reports using the 'html_report' formatter in your Cucumber.yml file.

JSON Reports: JSON reports can be generated to show the results of your Cucumber tests. These reports can be used to share the results of your tests with others or to integrate with other tools. You can generate JSON reports using the 'json_report' formatter in your Cucumber.yml file.

XML Reports: XML reports can be generated to show the results of your Cucumber tests. These reports can be used to share the results of your tests with others or to integrate with other tools. You can generate XML reports using the 'xml_report' formatter in your Cucumber.yml file.

When writing BDD tests, it's essential to follow certain best practices to ensure the effectiveness and efficiency of the tests.

1. Write Test Scenarios as early as possible:

It's a good practice if we are ready to write test scenarios before creating test code. Developing scenarios early helps you define software behavior and understand possible problems in a better way. Once a scenario is ready and reviewed by team members who agree with it, we can start implementing the functionality. This will save time and effort and take away uncertainty later, during the development, or when your team will run acceptance tests.

2. Features and Scenarios:

A feature is a product functionality, whereas a scenario is the specification of behavior.

Feature file:

- Feature file should be specific to the page, meaning if you have to cover the login scenario we should create a "login.feature"
- A single feature file can contain multiple scenarios
- Avoid mixing feature files
- Every feature should be ready to be executed alone

Scenario:

- To describe the scenarios, Gherkin keywords are used: Given, When, Then, But and And during writing the scenario we have to make sure we should use keywords in an appropriate way.

See below an example of a poorly written scenario.	A better way to write the same above scenario with fewer lines is as follows. It is best to avoid details that don't contribute much.

test > features > featureFiles > test.feature > ...

```
1  Feature: User wants to login into the site with valid data and Invalid data
2
3  Scenario: As an existing user, I want to log in successfully
4    Given the user is on the Home page
5    When the user clicks on the login link on Home Page
6    And the user can see the login screen
7    And the user enters his username and password
8    And the user is able to click on the login button
9    Then the user is logged in successfully
10   And the successful login message is displayed
11
```

test > features > featureFiles > test.feature > ...

```
1  Feature: User wants to login into the site with valid data and Invalid data
2
3  Scenario: As an existing user, I want to log in successfully.
4    Given the user is on the Home page
5    When the user navigates to the Login page
6    And the user enters the username and password
7    Then the successful login message is displayed
8
```

3. Use Background:

It is always the best practice to put steps that are repeated in every scenario into the Background. The background step is run before every scenario.

Ensure that you don't put too many steps in there, as your scenarios may become harder to understand. Given below is an example.

test > features > featureFiles > platform > test.feature > ...

```
1   Feature: User wants to login into the site with valid data and Invalid data
2
3     Background:
4       Given user navigates to the Website
5
6     Scenario: Login as a new sign-up user with valid data
7       When user enters the user name
8       And user enters the password
9       And user clicks on the sign-in button
10      Then user should be logged-in successfully
11      But Log-in button is not present
12
13
```

4. Try to Re-Use Step Definitions:

It's best practice to reuse those step definitions that we are using frequently in various scenarios e.g "Given: User navigates to the Website" this can be one common step that we need in every scenario. So, we can reuse this step.

Reusing the steps improves the maintainability if just in case there's any change come we need to update in a single step. We have to make sure language/words should be consistent if we want to re-use the step.

In the below example step is the same the but word isn't consistent Click and clicks both are different words. So we have to take care of case sensitive to re-use the step.

Given ("Click on 'Sign In link' on the Home Page")

Given ("click on 'Sign In link' on the Home Page")

5. Use a Data Table:

It's recommended to use Data Table to store the data. We can give data as parameters within the step but once we have a large set of data then it's not feasible to relinquish all data as parameters. It's better to use a data table once we have a large set of data.

The data table is used to inject data at the step level. In the below code we have provided data for step "I entered valid credential".

test > features > featureFiles > platform > test.feature > ...

```
1   Feature: User wants to login into the site with valid data and Invalid data
2
3     Background:
4       Given User navigate to the Website
5
6     @SmokeTest
7     Scenario: Login as a new sign-up user with valid data
8       When User entered valid credential
9         | email                  | validpassword | title |
10        | test1234@yopmail.com   |         12345 | Home  |
11        | testmon123@yopmail.com |         12345 | Home  |
12        | testwed123@yopmail.com |         12345 | Home  |
13        | testfri123@yopmail.com |         12345 | Home  |
14      And User click on sign-in button
15      Then Validate the title after login
16
```

6. Use Scenario Outline:

Scenario outline injects the info at the scenario level rather than the step level. Scenario outline followed by the keyword.

test > features > featureFiles > test.feature > ...

```
1   Feature: User want to login into the site with valid and invalid data
2
3     Scenario Outline: Login Validation
4       Given User navigate to the Website
5       When User enter "<email>" and "<validpassword>" to login Page
6       And User click on sign-in button
7       Then Validate the "<title>" after login
8
9     Example:
10      | email                    | validpassword     |title  |
11      | qatubeupdate@yopmail.com | 12345             | Home  |
12
```

7. Use of Tags:

Using tags could be the best practice once we want to run a bunch of test cases. There will be the case that we don't want to execute all test cases in a single run or we would like to execute a group of the test cases. So the best practice is to configure the test case by putting Tags on it. They are marked with @ followed by some text.

For example, a group of smoke test cases and sanity tests is created. You can put a tag over the scenario like @SmokeTest, @SanityTest, @RegressionTest, etc.

```
test > features > featureFiles > platform > test.feature > ...
  1   Feature: User want to login into the site with valid and invalid data
  2
  3     Background:
  4       Given User navigate to the Website
  5
  6     @SmokeTest
  7     Scenario: Login as a new sign-up user with valid data
  8       When User entered a valid credential
  9         | email                      | validpassword |
 10         | testqaautomation@yopmail.com |         12345 |
 11       And User clicks on sign-in button
 12       Then Validate the title after login
 13
 14     @SanityTest
 15     Scenario: Login with invalid data by entering an invalid password
 16       When User entered an invalid credential
 17         | email                      | invalidpassword |
 18         | testqaautomation@yopmail.com |          123456 |
 19       And User clicks on the sign-in button
 20       Then Error message should display
 21         | errormessage          |
 22         | Authentication failed |
 23
```

8. Map your Scenario with the Requirement:

It's a best practice if we will provide the requirement number (#Jira story id) after the scenario in the feature file. If any scenario fails, then we can track which requirement is fail from the execution report (See below test case execution screenshot).

For example, in lines #7, and #15 of the code below, the required number is specified for better tracking.

```
test > features > featureFiles > platform > test.feature > ...
  1   Feature: User want to login into the site with valid and invalid data
  2
  3     Background:
  4       Given User navigate to the Website
  5
  6     @SmokeTest
  7     Scenario: Login as a new sign-up user with valid data: QA-135, QA-156
  8       When User entered a valid credential
  9         | email                      | validpassword |
 10         | testqaautomation@yopmail.com |         12345 |
 11       And User clicks on sign-in button
 12       Then Validate the title after login
 13
 14     @SanityTest
 15     Scenario: Login with invalid data by entering an invalid password: QA-569, QA-281
 16       When User entered an invalid credential
 17         | email                      | invalidpassword |
 18         | testqaautomation@yopmail.com |          123456 |
 19       And User clicks on the sign-in button
 20       Then Error message should display
 21         | errormessage          |
 22         | Authentication failed |
 23
 24
```

9. Write in a declarative way, not Imperative:

Scenarios should be written in a way the user would describe them. Beware of scenarios that only describe clicking links and entering data in form fields, or of steps that contain code. This is not a feature description. Declarative features are realistic, compendious, and contain highly maintainable steps.

Example of Declarative Scenario:	Example of Imperative Scenario:

test > features > featureFiles > platform > test.feature > ...

```
1  Feature: User want to login into the site with valid data
2
3  Scenario: Verify user login successfully
4      Given User Navigates to the Website
5      When User enters credentials
6      And User clicks on the sign-in button
7      Then Validate the title after login
8
```

test > features > featureFiles > platform > test.feature > ...

```
1   Feature: User want to login into the site with valid data
2
3   Scenario: Verify login
4     Given I navigate to the Website
5     When I enter "username"
6     When I enter "password"
7     When I check the "Remember me" check box
8     When I clicks on the sign-in button
9     Then Validate the title after login
10
```

Project Structure

css Copy code

```
src
 ├── main
 │    └── java
 │         └── pages
 │              └── SignupPage.java
 │              └── LoginPage.java
 │              └── ProductPage.java
 │              └── CartPage.java
 └── test
      ├── java
      │    ├── runners
      │    │    └── TestRunner.java
      │    └── stepDefinitions
      │         └── SignupSteps.java
      │         └── LoginSteps.java
      │         └── AddToCartSteps.java
      │         └── PurchaseSteps.java
      │         └── LogoutSteps.java
      └── resources
           └── features
                └── signup.feature
                └── login.feature
                └── add_to_cart.feature
                └── purchase.feature
                └── logout.feature
```

****

Version Control System

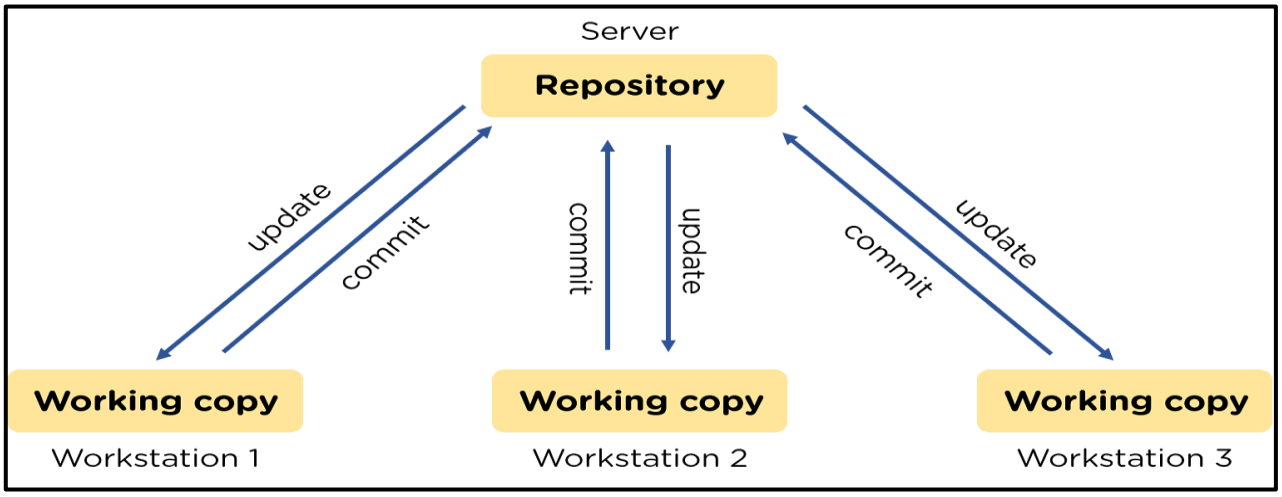
Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time. As development environments have accelerated, version control systems help software teams work faster and smarter. Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

Software developers working in teams are continually writing new source code and changing existing source code. The code for a project, app or software component is typically organized in a folder structure or "file tree". One developer on the team may be working on a new feature while another developer fixes an unrelated bug by changing code, each developer may make their changes in several parts of the file tree. Version control helps teams solve these kinds of problems, tracking every individual change by each contributor and helping prevent concurrent work from conflicting. Changes made in one part of the software can be incompatible with those made by another developer working at the same time. This problem should be discovered and solved in an orderly manner without blocking the work of the rest of the team. Further, in all software development, any change can introduce new bugs on its own and new software can't be trusted until it's tested. So testing and development proceed together until a new version is ready.

The responsibility of the Version control system is to keep all the team members on the same page. It makes sure that everyone on the team is working on the latest version of the file and, most importantly, makes sure that all these people can work simultaneously on the same project.

Let's try to understand the process with the help of this diagram:

There are 3 workstations or three different developers at three other locations, and there's one repository acting as a server. The work stations are using that repository either for the process of committing or updating the tasks.



Benefits of Version Control:

- **Managing and Protecting the Source Code:**

The Version Control System helps manage the source code for the software team by keeping track of all the code modifications. It also protects the source code from any unintended human error and consequences.

- **Keeping Track of All the Modifications Made to the Code:**

The team working on the project continuously produces new source codes and keeps making amendments to the existing code. These changes are recorded for future references and can be used if ever needed in the future to discover the root cause of any particular problem.

- **Comparing Earlier Versions of the Code:**

Since all the versions of the code are saved, this makes it possible for developers to go back at any time and compare the earlier versions of the code to help fix the mistake while reducing disruption to all team members.

- **Supports the Developers' Workflow and Not any Rigid Way of Working:**

Any suitable Version Control software will not impose any particular way of working. The Version Control Systems are known to provide a smooth and continuous flow of changes made to the code and prevent developers from getting frustrated in this clumsy mechanism.

- **Error recovery:**

VCS allows developers to revert to previous versions if new changes cause errors.

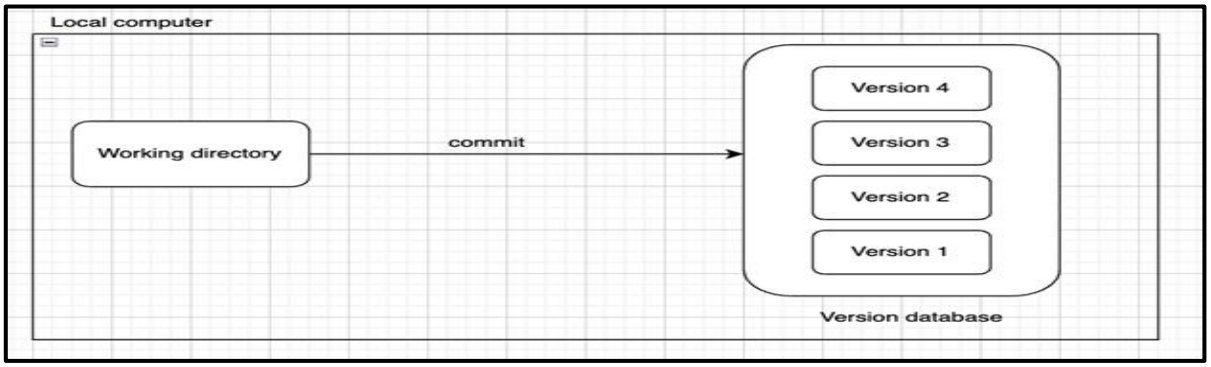
- **Backup:**

VCS acts as a backup of the codebase, making it easy to recover from failures like hard drive issues.

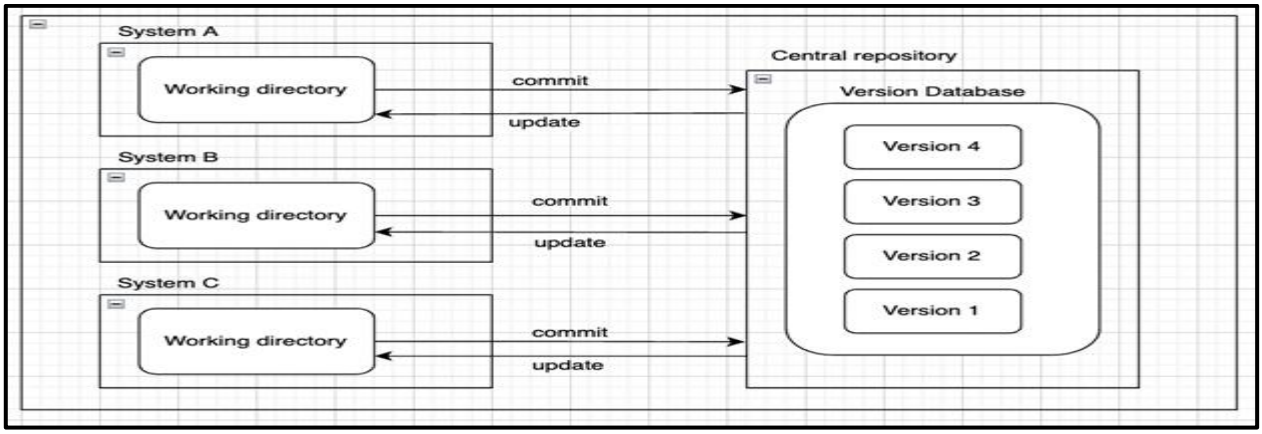
Types of Version Control Systems (VCS):

1. Manual Version Control (Copying Files)	**2. Local Version Control Systems**
3. Centralized Version Control Systems	**4. Distributed Version Control Systems**

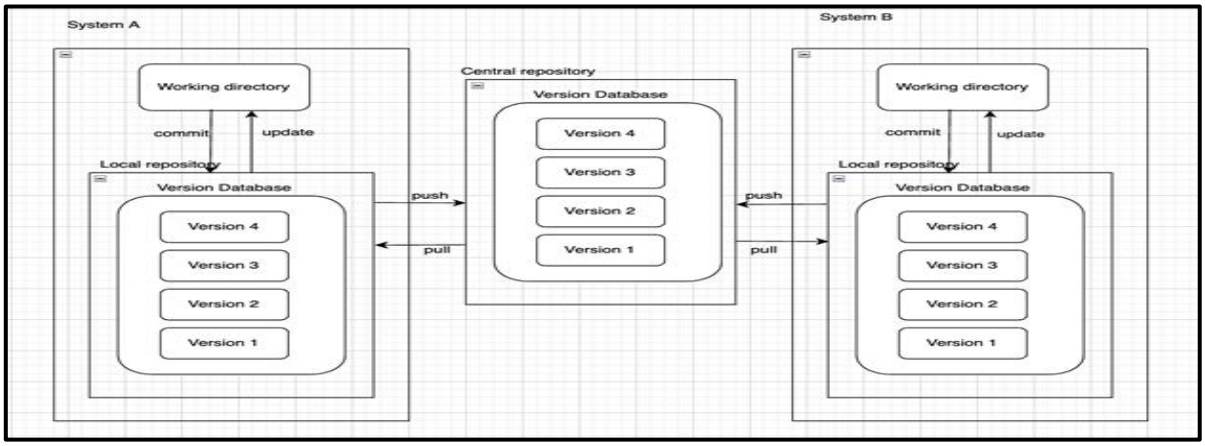
1. **Manual Version Control (Copying Files):** This is the most basic method where developers manually copy files to a new location with a descriptive name to track different versions, often used for small projects with minimal collaboration. Files are frequently copied to new directories, perhaps with timestamps added for organization.
2. **Local Version Control Systems (LVCS):** A system where all version tracking happens on a single developer's computer, with no central repository. A local version control system is a database situated on your local computer where every file change is stored as a patch. Each patch set contains only the changes made to the file since its last version. The purpose of a local version control system, or VCS, was to overcome the drawbacks of manual techniques. The main drawback with this system is that everything is stored locally. If anything were to happen to the local database, all the patches would be lost.



3. **Centralized Version Control Systems (CVCS):** A system where all versions of a project are stored on a central server, with developers "checking out" files to work on and then "committing" changes back to the server. It is designed to address problems with collaboration. All project files are stored on the central server. Developer workstations connected to the server to retrieve files for editing are called clients. Users check out files, make changes, and check them back into the central repository. Enables collaboration by multiple developers.



4. **Distributed Version Control Systems (DVCS):** DVCSs distribute the project's history instead of relying on a single server. Each developer working on the project has a complete copy (clone) of all files and their entire change history on their local machine. A Distributed Version Control System allows each user to have a complete copy of the repository. Users can work independently and merge changes later. Enhances collaboration and allows offline work. In distributed version control systems, clients don’t merely check out the latest snapshot of files from the server; instead, they fully mirror the repository, encompassing its entire history. Consequently, each collaborator in a project possesses a local copy of the entire project, constituting their own local database with a complete history. In this model, if the server becomes unavailable or fails, any of the client repositories can transmit a copy of the project’s version to another client or restore it to the server when it becomes accessible. It suffices for one client to contain a correct copy, which can then be easily distributed further.


System A
Working directory
commit
update
Local repository
Version Database
Version 4
Version 3
Version 2
Version 1
push
pull
Central repository
Version Database
Version 4
Version 3
Version 2
Version 1
push
pull
System B
Working directory
commit
update
Local repository
Version Database
Version 4
Version 3
Version 2
Version 1

****

GIT

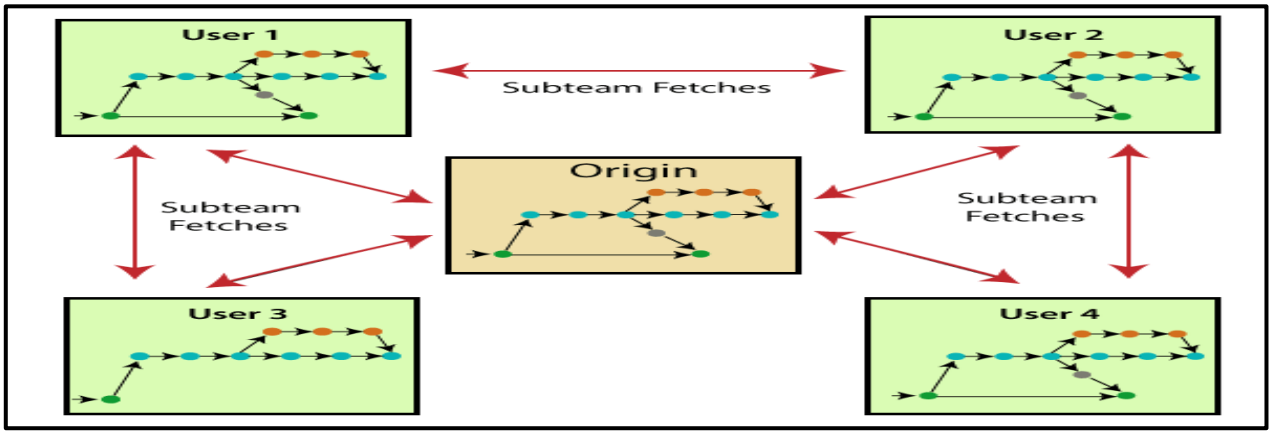
Git is a powerful version control system widely used for tracking changes in source code during software development. Created by Linus Torvalds in 2005, Git has become an essential tool for developers worldwide. Understanding Git can significantly enhance your coding efficiency and collaboration. Git is a modern and widely used distributed version control system [**DVCS**] in the world. It is developed to manage projects with high speed and efficiency. The version control system allows us to monitor and work together with our team members at the same workspace.

Git is an open-source distributed version control system. It is designed to handle minor to major projects with high speed and efficiency. It is developed to co-ordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace. There are many words to define git, but it is an open-source distributed version control system in simpler words. Git is foundation of many services like GitHub and GitLab, but we can use Git without using any other Git services. Git can be used privately and publicly.

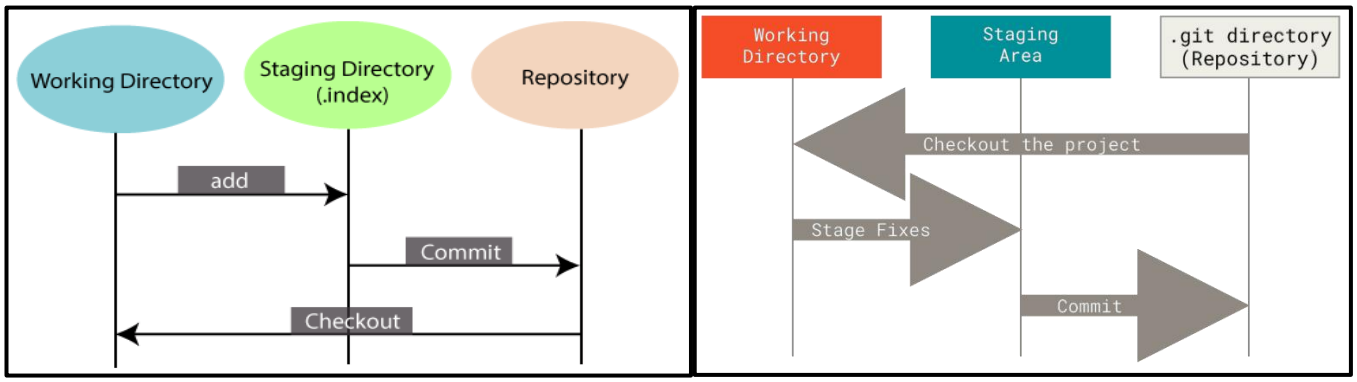
Why Git is Need: When a team works on real-life projects, git helps ensure no code conflicts between the developers. Furthermore, the project requirements change often. So a git manages all the versions. If needed, we can also go back to the original code. The concept of branching allows several projects to run in the same codebase.

Features of Git:

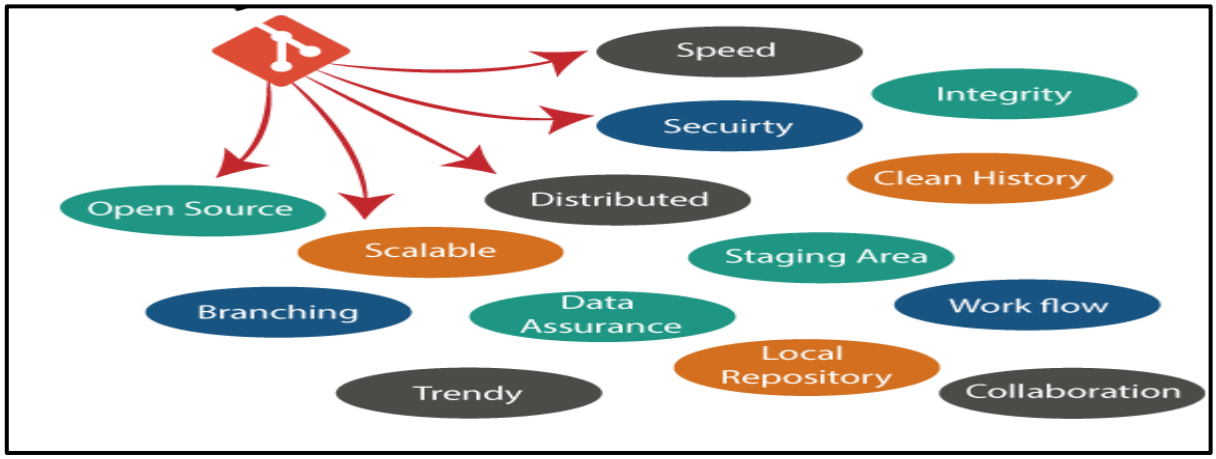
- **Open Source:** Git is an open-source tool. It is released under the GPL (General Public License) license.
- **Scalable:** Git is scalable, which means when the number of users increases, the Git can easily handle such situations.
- **Distributed:** One of Git's great features is that it is distributed. Distributed means that instead of switching the project to another machine, we can create a "clone" of the entire repository. Also, instead of just having one central repository that you send changes to, every user has their own repository that contains the entire commit history of the project. We do not need to connect to the remote repository; the change is just stored on our local repository. If necessary, we can push these changes to a remote repository.



- **Speed**: Git is very fast, so it can complete all the tasks in a while. Most of the git operations are done on the local repository, so it provides a huge speed. Also, a centralized version control system continually communicates with a server somewhere. Performance tests conducted by Mozilla showed that it was extremely fast compared to other VCSs.
- **Branching and Merging:** Branching and merging are the great features of Git, which makes it different from the other SCM tools. Git allows the creation of multiple branches without affecting each other. We can perform tasks like creation, deletion, and merging on branches, and these tasks take a few seconds only.
- **Staging Area:** The Staging area is also a unique functionality of Git. It can be considered as a preview of our next commit, moreover, an intermediate area where commits can be formatted and reviewed before completion. When you make a commit, Git takes changes that are in the staging area and make them as a new commit.



- **Maintain the clean History:** Git facilitates with Git Rebase; It is one of the most helpful features of Git. It fetches the latest commits from the master branch and puts our code on top of that. Thus, it maintains a clean history of the project.



Benefits of Git:

A version control application allows us to keep track of all the changes that we make in the files of our project. Every time we make changes in files of an existing project, we can push those changes to a repository. Other developers are allowed to pull your changes from the repository and continue to work with the updates that you added to the project files.

- **Saves Time:** Git is lightning-fast technology. Each command takes only a few seconds to execute so we can save a lot of time as compared to login to a GitHub account and find out its features.
- **Offline Working:** One of the most important benefits of Git is that it supports offline working. If we are facing internet connectivity issues, it will not affect our work. In Git, we can do almost everything locally. Comparatively, other CVS like SVN is limited and prefer the connection with the central repository.
- **Undo Mistakes:** One additional benefit of Git is we can Undo mistakes. Sometimes the undo can be a saviour option for us. Git provides the undo option for almost everything.
- **Track the Changes:** Git facilitates with some exciting features such as Diff, Log, and Status, which allows us to track changes so we can check the status, compare our files or branches.

Careers with Git: Knowing Git opens up numerous career opportunities, as version control is essential in almost all software development, QA, and IT roles. Here are some careers where Git knowledge is a valuable, often necessary skill:

- ✓ Software Developer/Engineer
- ✓ DevOps Engineer
- ✓ QA Engineer/Software Tester
- ✓ Data Scientist/Engineer
- ✓ System Administrator/IT Specialist
- ✓ Product Manager/Technical Project Manager
- ✓ Machine Learning Engineer
- ✓ Technical Writer
- ✓ Full-Stack Developer and many more roles

Installing Git:

Before you start using Git, you have to make it available on your computer. Even if it's already installed, it's probably a good idea to update to the latest version. You can either install it as a package or via another installer, or download the source code and compile it yourself. There are also a few ways to install Git on Windows.

Step#1: Download the Installer

The most official build is available for download on the Git website. Just go to " https://git-scm.com/downloads/win " win and the download will start automatically. After the download is complete, run the .exe file.

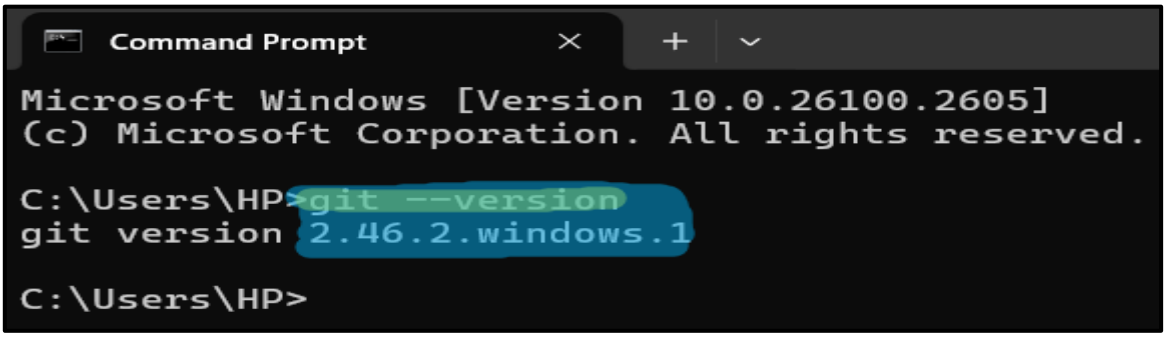
Step#2: Select Editor & Adjust Path

Follow the prompts in the setup wizard. Most default settings will be fine for general use, but here are some key steps:

- Firstly, the installer will ask which text editor to use for Git. You can choose from options like Vim, Notepad++, or Visual Studio Code.
- Make sure you choose the option that adds Git to your system PATH (recommended for ease of use in the command line).
- Select the default option (OpenSSL) to allow Git to communicate securely over HTTPS.
- The default choice, "Checkout Windows-style, commit Unix-style line endings," is ideal for most developers working in Windows environments.

Step#3: Complete the Installation

After configuring the setup, click "Install" and allow the installation to complete. Once done, you can launch Git Bash (installed with Git) or use Git from the Command Prompt (cmd) by typing the following command. **git --version**



GIT Commands:

1. **git config:** The git config command is used to set configuration options for your Git installation. These configurations can be at the system level (applying to all users and repositories on the system), global level (applying to all repositories for the current user), or local level (specific to a single repository). Common configurations include setting the user's name and email, which are associated with commits.

Syntax:
git config --global user.name "Your Name" # Set global username git config --global user.email "your.email@example.com" # Set global email git config --list # Show all configured settings git config --global core.editor "vim" # Set default text editor git config --global alias.co checkout # Create an alias for checkout
Example:
git config --global user.name "Krishna" git config --global user.email "Krishna@example.com" git config --list

2. **git init:** The git init command initializes a new Git repository in the specified directory. If no directory is specified, it initializes the repository in the current directory. This command sets up all the necessary files and directories that Git uses to track changes.

Syntax:
git init
Example:
mkdir my-project cd my-project git init

3. **git add:** The git add command adds changes from the working directory to the staging area. This prepares the changes to be included in the next commit. It's important to note that git add doesn't affect the repository until a commit is made.

Syntax:

```
git add <file>  # Add a specific file
git add .  # Add all changes in the current directory
git add --all  # Add all modified, deleted, and new files
```

Example:

```
git add index.html git add .
```

4. **git diff:** The git diff command displays the differences between various Git data sources, such as the working directory, staging area, and commits. It's useful for reviewing changes before staging or committing them.

Syntax:

```
git diff  # Show unstaged changes
git diff --staged  # Show changes that are staged
git diff <commit1> <commit2>  # Show differences between two commits
```

Example:

```
git diff index.html
```

5. **git commit:** The git commit command captures a snapshot of the project's currently staged changes. Commits are the building blocks of a Git repository, recording the history of changes.

Syntax:

```
git commit -m "Commit message"  # Commit staged changes with a message
git commit -a -m "Commit message"  # Commit all modified and deleted files
git commit --amend -m "Updated commit message"  # Modify the last commit message
```

Example:

```
git commit -m "Added new feature"
```

6. **git reset:** The git reset command is used to undo changes by resetting the current branch to a previous state. It helps to unstage files, remove commits, or even delete changes from the working directory.

Syntax:

```
git reset <file>  # Unstage a specific file
git reset --soft <commit>  # Reset to a commit but keep changes staged
git reset --mixed <commit>  # Reset to a commit and unstage changes
git reset --hard <commit>  # Reset to a commit and discard all changes
```

Example:

```
git reset HEAD~1  # Undo the last commit but keep changes
git reset --hard HEAD~1  # Undo the last commit and discard changes
```

7. **git status:** The git status command is used to display the state of the working directory and the staging area. It helps developers track which files have been modified, staged, or are untracked.

Syntax:

git status
Example: git status

8. **git merge:** The git merge command is used to combine changes from one branch into another. It integrates modifications from a specified branch into the current branch, preserving history and commit integrity.

Syntax: git merge <branch> # Merge a branch into the current branch git merge --no-ff <branch> # Merge with a commit history
Example: git merge feature-branch

9. **git push:** The git push command uploads local commits from your Git repository to a remote repository. It ensures that changes made locally are available to other collaborators working on the same project.

Syntax: git push origin <branch> # Push changes to a specific branch git push -u origin <branch> # Push and track branch
Example: git push origin main

10. **git pull:** The git pull command is used to fetch and integrate changes from a remote repository into the local branch. It is a combination of two commands:

git fetch: Retrieves new changes from the remote repository but does not apply them.

git merge: Merges the fetched changes into the current branch. This command keeps your local branch up to date with the remote repository.

Syntax: git pull origin <branch>
Example: git pull origin main

****

GitHub

GitHub is a cloud software development platform. It is commonly used for saving files, tracking changes, and collaborating on development projects. In recent years, GitHub has become the most popular social platform for software development communities. Individuals can contribute to open-source projects and bug reports, discuss new projects and discover new tools.

GitHub is an online hosting service for Git repositories. GitHub is an American company. It hosts source code of your project in the form of different programming languages and keeps track of the various changes made by programmers. It offers both distributed version control and source code management (SCM) functionality of Git. It also facilitates with some collaboration features such as bug tracking, feature requests, task management for every project. GitHub also facilitates with many of its features, such as access control and collaboration.

GitHub is a Git repository hosting service. It provides a Web-based graphical interface. Imagine working on a project at home and while you are away, maybe at a friend's place, you suddenly remember the solution to a code error that has kept you restless for days. You cannot make these changes because your PC is not with you. But if you have your project hosted on GitHub, you can access and download that project with a command on whatever computer you have access to. Then you can make your changes and push the latest version back to GitHub. GitHub lets you store your repo on their platform. Another awesome feature that comes with GitHub is the ability to collaborate with other developers from any location. Now that we have created and initialized our project locally, let's push it to GitHub.

Features of GitHub:

- **Open-source**: GitHub provides a complete ecosystem for open-source projects. You can sponsor maintainers, contribute to a project, use the open-source tool in your existing project, and promote your work.
- **Community Collaboration:** GitHub has become a community platform where issues, feature requests, code, and documentation contributions can be discussed.
- **Explore:** GitHub Explore tab helps you discover new projects, trending tools, and developer events.
- **GitHub Gists:** You can share the snippet of your code or embed it in a blog or website.
- **GitHub CLI:** It allows you to perform merge requests, review code, check issues, and monitor progress from the command line program.
- **Free Storage:** unlimited private and public repositories storage.
- **Web hosting:** You can publish your portfolio site or documentation. GitHub pages provide easy to build and deploy website experience.
- **CodeSpace:** a cloud development environment integrated with your GitHub repository.
- **Project:** a customizable, flexible tool for planning and tracking the work on GitHub.

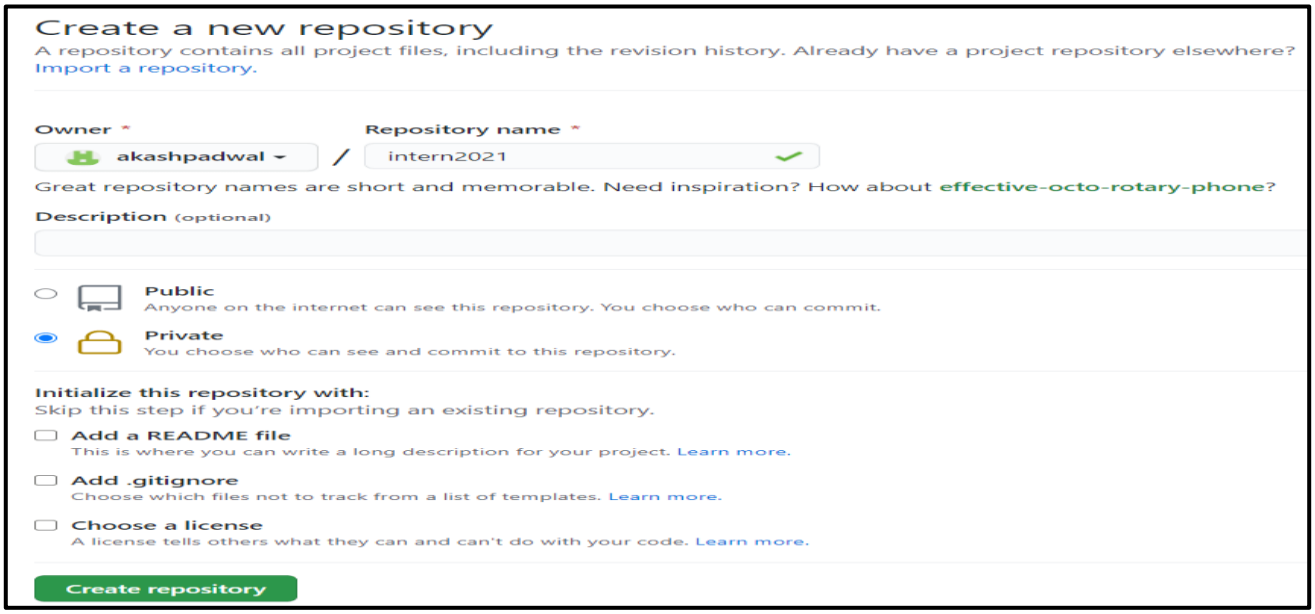
- **Automation:** GitHub Action automates development workflow such as build, test, publish, release, and deployment.

Create a new GitHub Account:

→ Go to GitHub in a web browser: " https://github.com "
" https://github.com/signup?source=form-home-signup&user_email "
→ Select Sign up
→ Enter your email address, username, and password
→ Select Continue
→ Verify your account by solving a puzzle
→ Enter the launch code sent to your email address
→ Select Create account
→ Verify your email address

How to Create New Repository in GitHub:

In the top right corner of GitHub's home page, we can observe a add (+) button, we just have click on that and select "New Repository" option. Give a name for your repository for Ex. "SeleniumScripts2025". Write a short description about your repository, Let the access be Private or Public. Below you can see "Initialize this repository with:", Select the first option Add a README file, now click on Create repository. Our repository is created.



****

CI/CD

Continuous Integration and Continuous Delivery (CI/CD) is a software development approach that aims to improve the speed, efficiency, and reliability of software delivery. This approach involves frequent code integration, automated testing, and continuous deployment of software changes to production.

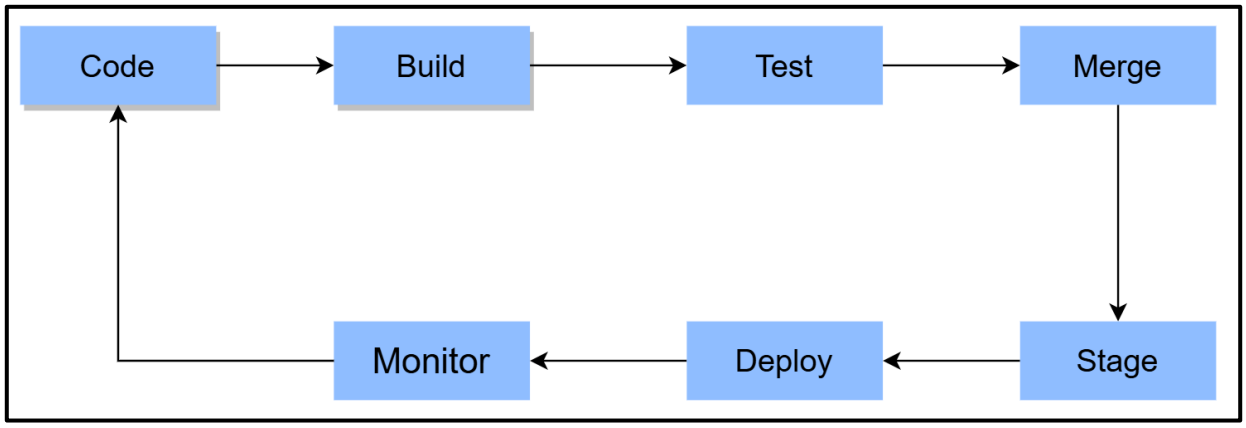
Before the adoption of CI/CD in the software development industry, the common approach was a traditional, waterfall model of software development. In this approach, each stage of the software development life cycle completed in sequence. The process typically involved gathering requirements, designing the software, coding, testing, and deployment.

The disadvantages of this traditional approach include: Slow-Release Cycles, High Failure Rates, Limited Collaboration, High Cost.

CI/CD emerged as a solution to these disadvantages, by introducing a more agile and collaborative approach to software development. CI/CD enables teams to work together, integrating their code changes frequently, and automating the testing and deployment process.

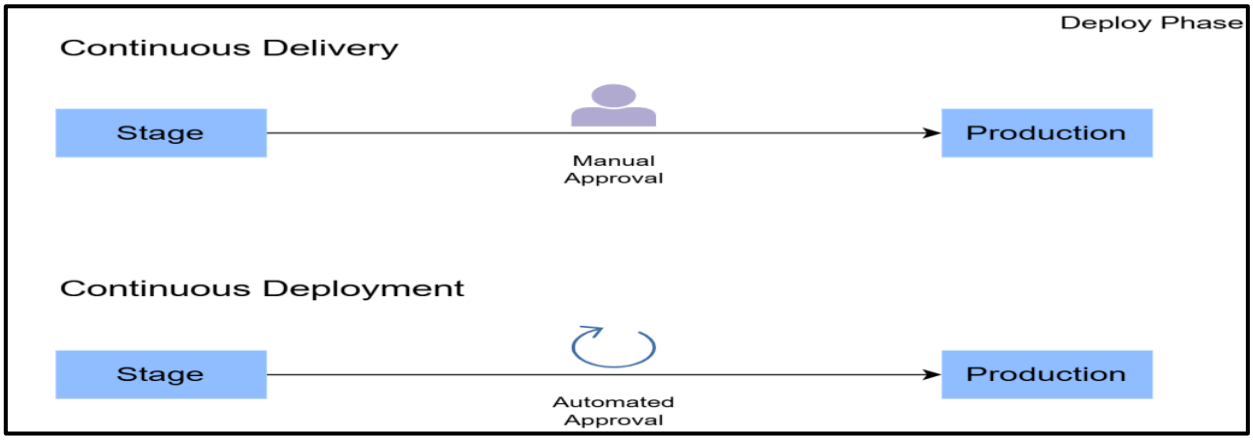
A typical software project spans multiple phases. In the development phase, the developer starts producing the source code and checks into a source code repository. In the build phase, a build management tool like Jenkins builds the checked-in code. It generates the build artifacts that are uploaded after thorough testing. The QA team will perform functional and integration tests in the test phase and ensure that the application code addresses all the business requirements.

Typically, the development happens in a branch separate from the main branch. After the successful testing, the tested code is merged with the main branch and uploaded to a staging environment. Automated testing is triggered after every source commits, and changes are merged back to the main branch. This is the beginning of a successful continuous integration implementation.



In the deployment phase, the deploy tool or a script developed by the admin will push the tested artifacts to production. Some companies set up a manual approval phase before pushing the code to production, and some perform automated deployments directly to production. Deployment that requires manual approval is called **continuous delivery**. It is called **continuous deployment** if an automated process is followed from code commit to

production deployment. So, the manual approval process is the only differentiating factor between continuous delivery and continuous deployment.



In the monitoring phase, the customer support team will actively monitor the deployed software and alert the production support team if they observe any issues.

How CI/CD Works:

CI/CD is an automated process that involves frequent code integration, automated testing, and continuous deployment of software changes to production.

A) Code Integration:

The first step in the CI/CD pipeline is code integration. In this step, developers commit their code changes to a remote repository (like GitHub, GitLab or BitBucket), where the code is integrated with the main codebase. This step aims to ensure that the code changes are compatible with the rest of the codebase and do not break the build.

B) Automated Testing:

Once the code is integrated, the next step is automated testing. Automated testing involves running a suite of tests to ensure that the code changes are functional, meet the expected quality standards, and are free of defects. This step helps identify issues early in the development process, allowing developers to fix them quickly and efficiently.

C) Continuous Deployment:

After the code changes pass the automated testing step, the next step is continuous deployment. In this step, the code changes are automatically deployed to a staging environment for further testing. This step aims to ensure that the software is continuously updated with the latest code changes, delivering new features and functionality to users quickly and efficiently.

D) Production Deployment:

The final step in the CI/CD pipeline is production deployment. In this step, the software changes are released to end-users. This step involves monitoring the production environment, ensuring that the software is running smoothly, and identifying and fixing any issues that arise.

The four steps of a CI/CD pipeline work together to ensure that software changes are tested, integrated, and deployed to production automatically. This automation helps to reduce errors, increase efficiency, and improve the overall quality of the software.

Benefits of CI/CD:

- **Faster Release Cycles:** By automating the testing and deployment process, CI/CD enables teams to release software more frequently, responding quickly to customer needs.
- **Improved Quality:** Automated testing ensures that software changes do not introduce new bugs or issues, improving the overall quality of the software.
- **Increased Collaboration:** Frequent code integration and testing require developers to work closely together, leading to better collaboration and communication.
- **Reduced Risk:** Continuous deployment allows developers to identify and fix issues quickly, reducing the risk of major failures and downtime.
- **Cost-Effective:** CI/CD reduces the amount of manual work required to deploy software changes, saving time and reducing costs.

****

Jenkins

There are several tools available for implementing CI/CD pipelines in software development. Each tool has its unique features. Jenkins is some of the most commonly used tools in CI/CD pipelines today.

Jenkins is a continuous integration open-source tool. It has been developed using Java. This allows real-time monitoring and recording of discrete improvements to a more comprehensive codebase. It lets developers easily identify and fix bugs in their codebase and simplify the validation of their builds. Jenkins is one of the best CI/CD tools because it closely monitors repetitive jobs and assists in automated execution during a project's production. Jenkins builds and tests our software projects which continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build.

It is a cloud-based system that can work as a standalone application running on a server. Alternatively, it can utilize a web server like Glassfish, JBoss, or WebSphere. Using Jenkins, developers can speed up their product creation cycle because Jenkins can simplify the build and test instantly. Jenkins promotes the entire software development life cycle (SDLC) by designing, testing, monitoring, deploying, and other cycle phases. Jenkins is a widely used application around the world that has around 300k installations and growing day by day.

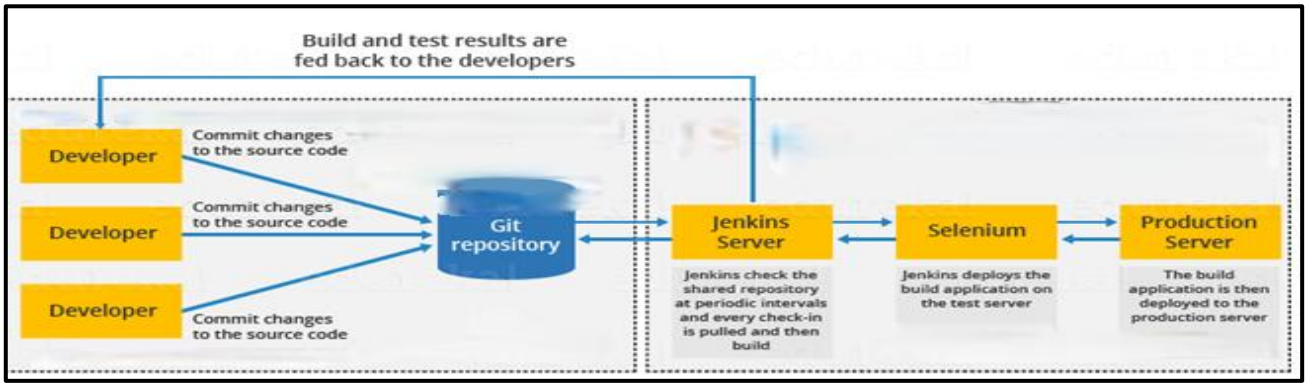
History of Jenkins:

Kohsuke Kawaguchi, who is a Java developer, working at SUN Microsystems, was tired of building the code and fixing errors repetitively. In 2004, he created an automation server called Hudson that automates build and test task.

In 2011, Oracle who owned Sun Microsystems had a dispute with Hudson open-source community, so they forked Hudson and renamed it as Jenkins.

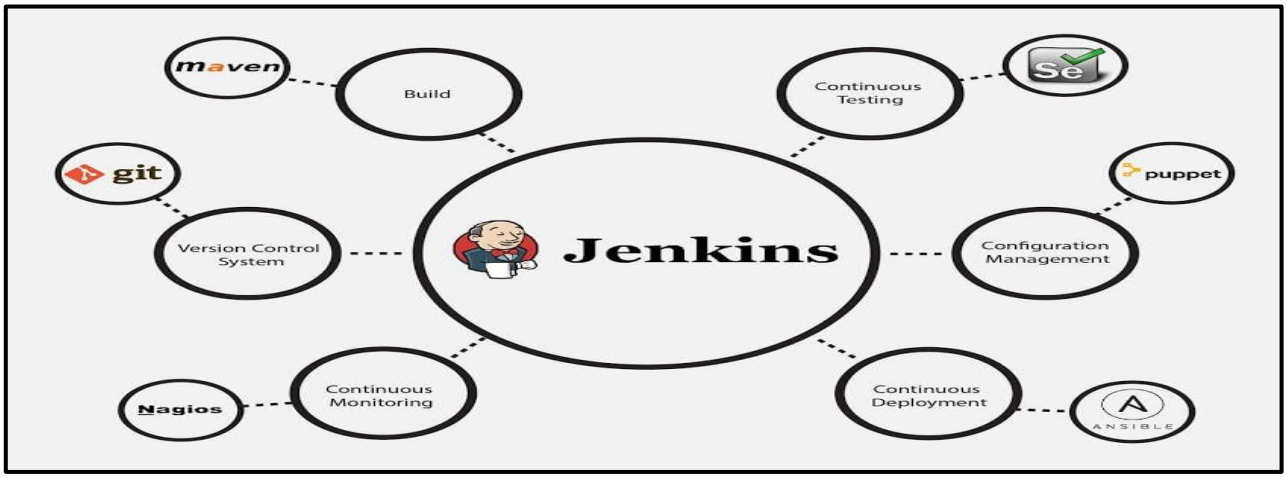
Both Hudson and Jenkins continued to operate independently. But in short span of time, Jenkins acquired a lot of contributors and projects while Hudson remained with only 32 projects. Then with time, Jenkins became more popular, and Hudson is not maintained anymore.

Jenkins is the most mature Continuous Integration tool available so let us see how Continuous Integration with Jenkins overcame the above shortcomings.



Let's see how Jenkins works. The above diagram is representing the following functions:

- First of all, a developer commits the code to the source code repository. Meanwhile, the Jenkins checks the repository at regular intervals for changes.
- Soon after a commit occurs, the Jenkins server finds the changes that have occurred in the source code repository. Jenkins will draw those changes and will start preparing a new build.
- If the build fails, then the concerned team will be notified.
- If built is successful, then Jenkins server deploys the built in the test server.
- After testing, Jenkins server generates feedback and then notifies the developers about the build and test results.
- It will continue to verify the source code repository for changes made in the source code and the whole process keeps on repeating.



Advantages of Jenkins:

- It is an open-source tool.
- It is free of cost.
- It does not require additional installations or components. Means it is easy to install.
- Easily configurable.
- It supports 1000 or more plugins to ease your work. If a plugin does not exist, you can write the script for it and share with community.
- It is built in java and hence it is portable.

- It is platform independent. It is available for all platforms and different operating systems. Like OS X, Windows or Linux.
- Easy support, since its open source and widely used.
- Jenkins also supports cloud-based architecture so that we can deploy Jenkins in cloud-based platforms.

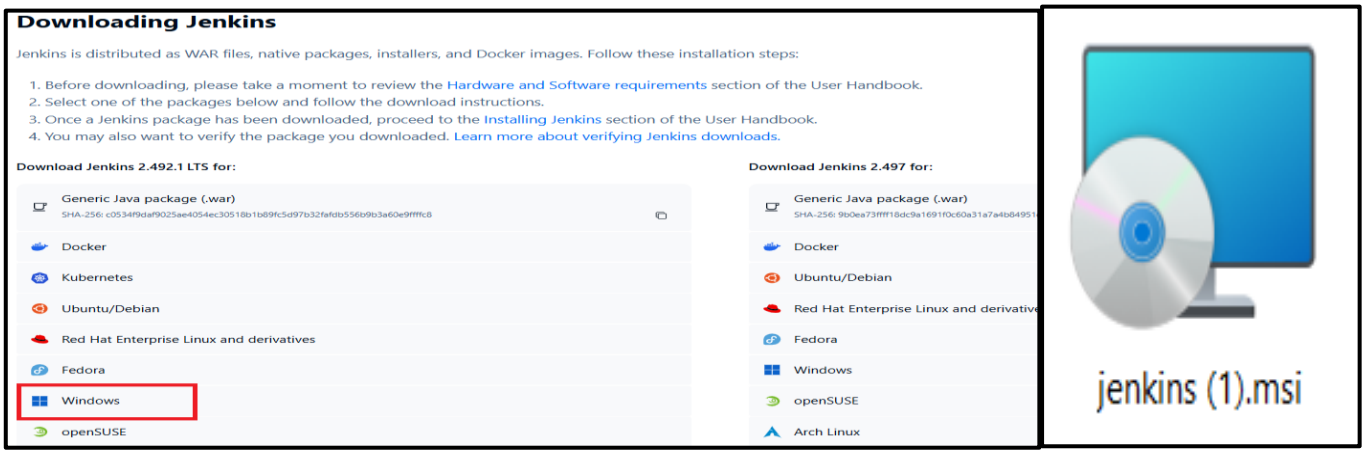
Steps to Install Jenkins on Windows:

Jenkins is typically run as a standalone application in its own process. The Jenkins WAR file bundles Winstone, a Jetty servlet container wrapper, and can be started on any operating system or platform with a version of Java supported by Jenkins.

Follow the below step to Download and Install Jenkins in your Windows operating system:

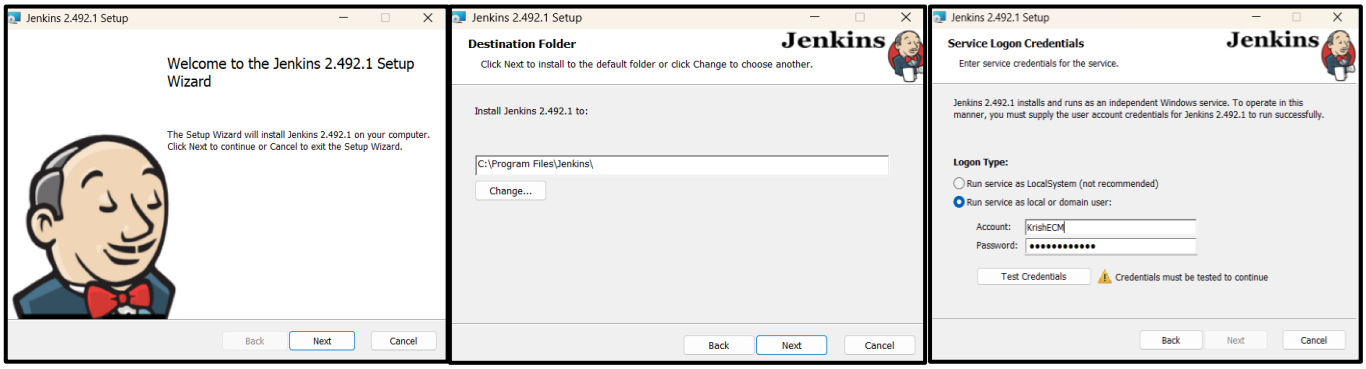
Step#1: Go to "https://jenkins.io/download/"

Step#2: Choose the Platform Windows. For this we selected Windows, you can download either LTS or weekly releases. Click on Download for Windows 64-bit. File will download



Step#3: Once the download is complete, run the jenkins.msi installation file. The setup wizard starts. Click Next to proceed. Select the install destination folder and click Next to continue.

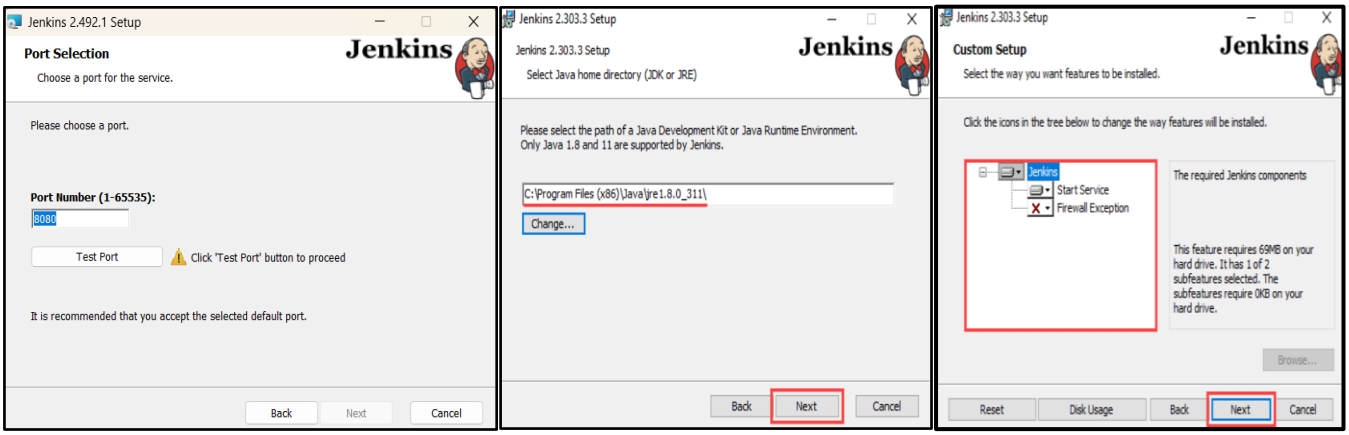
Step#4: Under the Run service as a local or domain user option, enter the domain username and password for the user account you want to run Jenkins with. Click Test Credentials to verify the login data, then click Next to proceed.



Step#5: Enter the port number you want Jenkins to run on. Click Test Port to check if the selected port is available, then click Next to continue.

Step#6: Select the directory where Java is installed on your system and click Next to proceed.

Step#7: Select the features you want to install with Jenkins and click Next to continue.



Step#8: Click Install to start the installation process. Once the installation is complete, click Finish to exit the install wizard.

Why we need Jenkins:

- **Automation**: Jenkins automates building, testing, and deployment tasks.
- **Continuous Integration (CI)**: Identifies and addresses integration issues early, ensuring higher code quality.
- **Continuous Delivery/Deployment (CD)**: Automates the deployment process for faster and more reliable releases.
- **Customization and Extensibility**: Highly customizable and extensible through a variety of plugins.
- **Monitoring and Reporting**: Provides real-time monitoring and detailed reports for better visibility.
- **Community Support**: Large and active community ensures continuous improvement and reliable usage.

****

Real time Selenium Interview Questions and Answers

✍ **What is Automation Testing?**

Ans: Automation testing is a process in which software tools execute pre-scripted tests on a software application before it is released into production. Special software is used to control the test execution, actual outcomes and predicted outcomes comparison, the test preconditions setting up, and other test control and test reporting functions.

Selenium Grid is a part of the Selenium Suite that specializes on running multiple tests across different browsers, operating systems, and machines in parallel.

✍ **What is the Limitation of Selenium?**

Ans-Some limitations of Selenium Automation tool are as follows:

1. It does not support and non-web-based applications, it only supports web-based applications.
2. You need to know at least one of the supported languages very well in order to automate your application successfully.
3. No inbuilt reporting capability so you need plugins like JUnit and TestNG for test reports.

✍ How many types of Webdriver APIs are available in Selenium?

The list of driver classes could be used for the browser automation.

 ChromeDriver, FirefoxDriver, EdgeDriver, SafariDriver , OperaDriver HtmlUnitDriver, InternetExplorerDriver, IOSDriver, RemoteWebDriver AndroidDriver,

✍ **Have you worked on different browsers? Have you ever performed Cross browser testing?**

Ans: Yes, I have worked on different browsers like Mozilla, Chrome, and Edge performed cross browser testing demo. Every month a new browser is coming into market and it became very important to test our web application on different browser. Selenium supports Cross browser testing.

✍ **How can we launch different browsers in Selenium WebDriver?**

We should create an instance of a driver of a particular browser:

```
WebDriver driver = new FirefoxDriver();
WebDriver driver = new ChromeDriver();
WebDriver driver = new SafariDriver();
WebDriver driver = new EdgeDriver();
WebDriver driver = new InternetExplorerDriver();
```

✍ **How can we maximize browser window in Selenium?**

Ans: driver.manage().window().maximize(); //command is used to maximize browser window in Selenium

✍ **What kinds of test types are supported by Selenium?**

1. Functional Testing 2. Regression Testing 3. Sanity Testing 4. Smoke Testing 5. Responsive Testing 6. Cross Browser Testing 7. UI testing (black box) 8. Integration Testing

✍ **How many test cases you have automated per day?**

Ans- It totally depends on your manual test cases. Sometimes we can automate 3-4 test cases per day which contain limited operation and some validation. Some test cases may take 1 day or more than one day as well. It totally depends on test case complexity.

✍ **Give an example of the languages supported by WebDriver.**

Ans: Java, C#, Python,JavaScript and Ruby, are all supported directly by the development team. There are also PHP and Perl WebDriver implementations.

✍ **Diffrence between findElement and FindElements?**

Ans- findElement () will return only single WebElement and if that element is not located or we use some wrong selector then it will throw NoSuchElement exception.

WebElement element = driver.findElements(By.xpath("//div[@id='example']//ul//li"));

findElements() will return List of WebElements – for this we need to give locator in such a way that it can find multiple elements and will return you list of webelements then using List we can iterate and perform our operation.

List elementList = driver.findElements(By.xpath("//div[@id='example']//ul//li"));

✍ **Can we find all links on a web page?**

Ans: As all links are of anchor tag 'a', so we can find all of them on a web page by locating elements of tagName 'a':

List links = driver.findElements(By.tagName("a"));

System.out.println("All Links: "+ links);

✍ **Can Selenium handle Windows based pop-up?**

Ans: Windows pop-ups cannot be handled by using Selenium. Because it supports only web application testing.

✍ **What are different locators available in Selenium?**

Ans: There are 8 types of locators are available in selenium that are as follows:

id, name, ClassName, css-selector, TagName, linkText, partialLinkText and xpath.

✍ **What is the difference between / and // in XPATH?**

Ans- "/" It's starts search selection from root element in document. (absolute path) "//" It start selection from anywhere in XML document. (relative path)

✍ **Can you please explain XPATH and CSS technique? How to handle dynamic changing elements?**

Ans: CSS Selector: CSS mainly used to provide style rules for the web pages and we can use for identifying one or more elements in the web page using css. If you start using css selectors to identify elements, you will love the speed when compared with XPath. CSS selector is always the best possible way to locate complex elements in the page.

XPath Selector: XPath is designed to allow the navigation of XML documents, with the purpose of selecting individual elements, attributes, or some other part of an XML document for specific processing There are two types of xpath

1. Native Xpath, it is like directing the xpath to go in direct way. like

Example: html/head/body/table/tr/td

Here the advantage of specifying native path is, finding an element is very easy as we are mention the direct path. But if there is any change in the path (if some thing has been added/removed) then that xpath will break.

2. Relative Xpath. In relative xpath we will provide the relative path, it is like we will tell the xpath to find an element by telling the path in between. Advantage here is, if at all there is any change in the html that works fine, until unless that particular path has changed. Finding address will be quite difficult as it need to check each and every node to find that path. Example: //table/tr/td

✍ **How can the user get a text of a web element?**

Ans: User can retrieve the text of the specified web element by using get command. It doesn't require any parameter but returns a string value.

String Text = driver.findElement(By.id("Some Text")).getText();

✍ **How a text written in a text field could be cleared?**

Ans: A text written in a text field could be deleted by using the clear() method.

✍ **How to check a CheckBox in Selenium?**

Ans: The same click() method could be used for checking checkbox as well as for clicking buttons or radio buttons.

✍ **How to verify checkbox (any element) is enable/disabled/ checked/Unchecked/ displayed/ not displayed?**

Ans: We have different Boolean methods for enable / disable, checked / unchecked and displayed / not displayed that are as follows:

1.There's a method "isEnabled()", that checks whether a WebElement is enabled or not. You can use the below code to check for that; boolean enabled = driver.findElement(By.xpath("//xpath of the checkbox")).isEnabled();

2.To check whether the checkbox is checked/selected or not, you can use "isSelected()" method, which you can use like this; boolean checked = driver.findElement(By.xpath("//xpath of the checkbox")).isSelected();

✍ **Explain the meaning of assertion in Selenium and what are the types of assertion?**

Ans: Assertion is used as a verification point. It verifies that the application state conforms to the expectation. The types of assertion are "assert", "verify" and "waifFor".

✍ **Could cookies be deleted in Selenium?**

Ans: driver.manage().deleteAllCookies(); //command is used for deleting all cookies

✍ **How to perform right click using Selenium WebDriver?**

Ans: The next Actions class is used to perform right click:

Actions act = new Actions(driver); // where driver is WebDriver type

act.moveToElement(webElement).perform();

act.contextClick().perform();

✍ **How to work with dropdown?**

Ans- WebDriver's support classes called "Select", which provides useful methods for interacting with select options. User can perform operations on a select dropdown and also de-select operation using the below methods. We can select or deselect option in dropdown by using following methods.

Syntax: Select Se=new Select(element);

Se.selectByIndex(index);

Se.selectByvalue(value);

Se.selectByVisibleText(text);

We can also deselect the item using same thing that is just above method like.

✍ **Have you worked with Web table (Calendar)? If yes, then what was your approach.**

Ans- Yes. First need to analysis its web page html code for this element. To find which type of calendar is, then you can decide you can solve this calendar by using selenium Webdriver or using JavaScript executer. It all depends on the scenario the code. Now a day, there would be no. of new type of calendar are using by dev teams. We can't handle these by using selenium but using JavaScript executer we get solution.

✍ **Can you tell me some navigation commands?**

Ans: To access the navigation's method, just type driver.navigate().. The intelligence feature of eclipse will automatically display all the public methods of Navigate Interface.

Command - driver.navigate().to(appUrl);

It does exactly the same thing as the driver.get(appUrl) method. Where appUrl is the website address to load. It is best to use a fully qualified URL.

forward() : void – This method does the same operation as clicking on the Forward Button of any browser. It neither accepts nor returns anything.

Command - driver.navigate().forward();

Takes you forward by one page on the browser's history.

back() : void – This method does the same operation as clicking on the Back Button of any browser. It neither accepts nor returns anything.

Command - driver.navigate().back();

Takes youback by one page on the browser's history.

refresh() : void – This method Refresh the current page. It neither accepts nor returns anything.

Command - driver.navigate().refresh();

✍ **How do perform drag and drop using Selenium WebDriver?**

Ans: The next Actions class is used to perform drag and drop:

Actions builder = new Actions(driver);

Action dragAndDrop = builder.clickAndHold(SourceElement)

moveToElement(TargetElement)

release(TargetElement).build().dragAndDrop.perform();

✍ **How to check if an element is visible on the page?**

Ans: The return method type is logical. If it returns true then element is visible otherwise it is not. isDisplayed() method could be used for it:
driver.findElement(By.id("id_of_element")).isDisplayed();

✍ **How to check if a button is enabled on the page?**

An: isEnabled() method could be used for it:
driver.findElement(By.id("id_of_element")).isEnabled();

✍ **Can you write the code to double click an element in Selenium?**

Ans: Code to double click an element in Selenium:

```
Actions action = new Actions(driver);
WebElement element=driver.findElement(By.id("elementId"));
action.doubleClick(element).perform();
```

✍ **How to mouse hover an element in Selenium?**

Ans: Code to mouse hover over an element in Selenium:

```
Actions action = new Actions(driver);
WebElement element=driver.findElement(By.id("elementId"));
action.moveToElement(element).perform();
```

✍ **What kind of keyboard operations can be performed in Selenium?**

Ans: Selenium lets to perform different kinds of keyboard operations, such as:
 .pressKey("non-text keys")(KeyDown) is used for keys like control, function keys etc that are non-text
 .releaseKey("non-text keys")(KeyUp) is used in conjunction with key press event to simulate releasing a key from keyboard event
 .sendKeys("sequence of characters") is used for passing character sequence to an input or textbox element.

✍ **Difference between Quit and Close?**

Ans: driver.close and driver.quit are two different methods for closing the browser session in Selenium WebDriver.
o driver.close – It closes the the browser window on which the focus is set.
o driver.quit – It basically calls driver.dispose method which in turn closes all the browser windows and ends the WebDriver session gracefully. You should use driver.quit whenever you want to end the program. It will close all opened browser window and terminates the WebDriver session. If you do not use driver.quit at the end of program, WebDriver session will not close properly and files would not be cleared off memory. This may result in memory leak errors.

✍ **What is Page Load Timeout?**

Ans: When automation script run on the browser. Sometimes scripts are faster than the web application that time scripts looking for an element but it can't be found because the web page not loaded completely and throws an element not found or element not visible exception. To eliminate these kind of exception and ensuring script run smoothly for this we

mention or set page load time out. Ex: driver.manage().timeouts().pageLoadTimeout(10, TimeUnit.SECONDS);

✍ **Can you explain implicit wait, explicit wait and fluent wait?**

Ans- Implicit Wait: Selenium WebDriver has borrowed the idea of implicit waits from Watir. This means that we can tell Selenium that we would like it to wait for a certain amount of time before throwing an exception that it cannot find the element on the page. We should note that implicit waits will be in place for the entire time the browser is open. This means that any search for elements on the page could take the time the implicit wait is set for.

Ex: driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

WebDriverWait wait = new WebDriverWait(driver, 10);

WebElement element = wait.until(ExpectedConditions.elementToBeClickable(By.id("someid")));

Fluent Wait: Each FluentWait instance defines the maximum amount of time to wait for a condition, as well as the frequency with which to check the condition. Furthermore, the user may configure the wait to ignore specific types of exceptions whilst waiting, such as NoSuchElementExceptions when searching for an element on the page.

Wait wait = new FluentWait(driver);

✍ **Write a code to wait for a particular element to be visible on a page. Write a code to wait for an alert to appear?**

We can write a code such that we specify the XPath of the web element that needs to be visible on the page and then ask the WebDriver to wait for a specified time. Look at the sample piece of code below:

1. WebDriverWait wait=new WebDriverWait(driver, 20);
2. Element = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("<xpath")));

Similarly, we can write another piece of code asking the WebDriver to wait until an error appears like this:

1. WebDriverWait wait=new WebDriverWait(driver, 20);
2. Element = wait.until(ExpectedConditions.alertIsPresent());

✍ **What is JavaScript Executor and where you have used JavaScript executor?**

Ans- JavascriptExecutor it is an interface. It Indicates that a driver can execute JavaScript, providing access to the mechanism to do so. There were lots of scenarios' their we need java-script should be executing for some element that are as follows: 1. when element is not clickable using locators then we can have used JavaScript.

2. While working with frames we used JavaScript.

3. The most recently while working with ck-editor I used JavaScript executers.

4. For the bootstrap calendar when the conditions for using selenium command it can't be possible that time I used JavaScript executer etc.

Syntax: JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript(Script,Arguments);

Examples: 1. How to generate Alert Pop window in selenium?
JavascriptExecutor js = (JavascriptExecutor)driver;
Js.executeScript("alert('hello world');");

✍ **How to capture Screenshot in Selenium? Can we capture screenshot only when test fails?**

Ans- For taking screenshots Selenium has provided TakesScreenShot interface in this interface you can use getScreenshotAs method which will capture the entire screenshot in form of file then using FileUtils we can copy screenshots from one location to another location.

Ex: // Take screenshot and store as a file format
File src= ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(src, new File("C:/selenium/error.png"));

✍ **How to scroll in Selenium Webdriver?**

Ans- Selenium support auto scroll to find an element but sometimes we need to scroll based on requirement like scroll up and down. We can perform this using Java Script. In this video, we will discuss How to Scroll page up and down in Selenium Webdriver. Ex: WebDriver driver = new FirefoxDriver(); JavascriptExecutor jse = (JavascriptExecutor)driver; jse.executeScript("window.scrollBy(0,250)", "");

✍ **How to upload files in Selenium? Have you ever used AutoIT?**

Ans-We can upload file in web application by using directly sending path in sendKeys. But at sometimes selenium path could not accept or upload the things, for this we used AutoIT tool. 1-AutoIt is freeware automation tool that can work with desktop application too. 2-It uses a combination of keystrokes, mouse movement and window/control manipulation in order to automate tasks in a way not possible or reliable with other languages (e.g. VBScript and SendKeys).

✍ **What is Actions class in Selenium (How to perform Mouse Hover, Keyboard events, DragAndDrop etc?)**

Ans-In Webdriver, handling keyboard events and mouse events (including actions such as Drag and Drop or clicking multiple elements With Control key) are done using the advanced user interactions API . It contains Actions and Action classes which are needed when performing these events. For all advance activity in Selenium Webdriver we can perform easily using Actions class like Drag and Drop, mouse hover, right click, clickandhold, releasemouse many more. We have predefined method called dragAndDrop(source, destination) which is method of Actions class.

To use mouse actions, we need to use current location of the element and then perform the action. The following are the regularly used mouse and keyboard events: Method :clickAndHold() Purpose: Clicks without releasing the current mouse location Method : contentClick() Purpose: Performs a context-click at the current mouse location.

Method: doubleClick() Purpose: Performs a double click at the current mouse location Method: dragAndDrop(source,target) Parameters: Source and Target Purpose: Performs

click and hold at the location of the source element and moves to the location of the target element then releases the mouse.
Method : dragAndDropBy(source,x-offset,y-offset) Parameters: Source, xOffset - horizontal move, y-Offset - vertical move Offset. Purpose: Performs click and hold at the location of the source element moves by a given off set, then releases the mouse.
Method: keyDown(modifier_key) Parameters: Modifier_key (keys.ALT or Keys.SHIFT or Keys.CONROL) Purpose: Performs a modifier key press, doesn't release the modifier key.
Method: keyUp(modifier_key) Parameters: Modifier_key (keys.ALT or Keys.SHIFT or Keys.CONROL) Purpose: Performs a key release.

✍ **What is Headless testing?**

Ans- A browser, which does not have any GUI it means which runs in background. If you run your programs in Firefox, Chrome, Edge and different browser then you can see how the browser is behaving but in headless browsers, you cannot.

```
// Declaring and initialize HtmlUnitWebDriver
WebDriver driver = new HtmlUnitDriver();
ChromeOptions options = new ChromeOptions();
options.addArguments("--headless");
FirefoxOptions options = new FirefoxOptions();
options.setHeadless(true);
EdgeOptions options = new EdgeOptions();
options.addArguments("--headless");
```

✍ **Have you created any framework? Which framework you have used?**

Ans. Yes. I have created Data-Driven Framework and added some functionalities of Hybrid framework and Page object model.

✍ **Which framework you have used and why?**

Ans- I have created Data-Driven Framework with some capabilities of Page object model framework. After analyzing condition of project I recommend myself to make data driven framework. And after sometime added some functionalities. As per what I need. And the most important thing is that it easier maintain and create.

✍ **What is POM (Page Object Model) and what is the need of it?**

Ans- Page Object Model Framework has now a days become very popular test automation framework in the industry and many companies are using it because of its easy test maintenance and reduces the duplication of code. The main advantage of Page Object Model is that if the UI changes for any page, it doesn't require us to change any tests, we just need to change only the code within the page objects (Only at one place). Many other tools which are using selenium, are following the page object model.

✍ **What are the challenges you have faced while automating your application?**

Ans- Challenges faced that are as follows:

 Frequently changing UI. It always need to make changes in code most of the time.

 Stuck somewhere while running automation scripts in chrome browser getting error that element is not visible, element not found.
 New kind of element like ck-editor, bootstrap calendar and dynamic web tables. But get the solution always.
 Reused of test scripts.

✍ **What could be the cause of Selenium WebDriver test to fail?**

Ans: There are some causes of Selenium WebDriver test to fail:
 SeleniumWebDriver element waiting to access did not appear on the web page and the operation timed out
 SeleniumWebDriver is trying to access not created element
 SeleniumWebDriver cannot locate the element, because the locator has been changed

✍ **What is a Data-driven framework?**

Ans: The Data Driven test design framework follows a design paradigm where test logic is fixed but varies the test data. The data itself can be in different repositories like a simple .csv file, .json file or .xls sheet, or database and can add the tests merely updating those external files or DB (instead of placing in test code itself).

✍ **What is a Keyword-driven framework?**

Ans: The keyword driven framework is a methodology where actions or steps are treated as keywords. These keywords (like click, move, type etc.,) are stored in some external repositories along just like data (in .csv/.json/.xls/DB).

✍ **What is the Hybrid framework?**

Ans: The combination of data driven and keyword driven framework is called the hybrid. Here the operations/instructions/keywords in a separate repository (.csv/.xls/.json/DB) and data is in separate (.csv/.xls/.json/db from data provider) and the tests/driver would read both and perform the actual tests automatically. In this design, we get the best of both methodologies, and it is kind of practical in most of the automation cases.

✍ **What are the different type of exception available in Selenium? Have you faced any exception while automation?**

Ans- Yes. I have faced lots of exception. List are as follows:
1. ElementNotSelectableException
2. ElementNotVisibleException
3. NoSuchAttributeException
4. NoSuchElementException
5. NoSuchFrameException
6. TimeoutException
7. Element not visible at this point

✍ **What is Alert window / JavaScript Alert and How to handle alert in Selenium Webdriver?**

Ans: There are two types of alerts that we would be focusing on majorly:
1. Windows based alert pop ups

2. Web based alert pop ups

As we know that handling windows based pop ups is beyond WebDriver's capabilities, thus we would exercise some third party utilities to handle window pop ups. Handling pop up is one of the most challenging piece of work to automate while testing web applications. Owing to the diversity in types of pop ups complexes the situation even more. Generally JavaScript popups are generated by web application and hence they can be easily controlled by the browser. Alert is an interface. There below are the methods that are used

```
Alert alert = driver.switchTo().alert();          //open alert, prompt or confirmation
alert.accept();                 //Will Click on OK button.
alert.dismiss();                // Will click on Cancel button.
alert.getText();                //will get the text which is present on the Alert.
alert.sendkeys();               //Will pass the text to the prompt popup
```

✍ **Have you ever worked on frames? In addition, how to handle frames in Selenium?**

Ans: Yes. In Selenium to work with iFrames, we have different ways to handle frame depending on the need. Please look at the below ways of handling frames.
driver.switchTo().frame(int arg0);

Select a frame by its (zero-based) index. That is, if a page has multiple frames (more than 1), the first frame would be at index "0", the second at index "1" and so on.

driver.switchTo().frame(WebElement frameElement);

✍ **How can you switch back from a frame?**

Ans: defaultContent() method is used to switch back from a frame.

✍ **What is TestNG?**

Ans: TestNG is a testing framework inspired from JUnit and NUnit, but introducing some new functionalities that make it more powerful and easier to use. TestNG is an open source automated testing framework; where NG means Next Generation. TestNG is similar to JUnit (especially JUnit 4), but it is not a JUnit extension. It is inspired by JUnit. TestNG Features Supports annotations.

1. TestNG uses more Java and OO features.
2. Supports testing integrated classes (e.g., by default, no need to create a new test class instance for every test method).
3. Separates compile-time test code from run-time configuration/data info.
4. Flexible runtime configuration.
5. Introduces 'test groups'. Once you have compiled your tests, you can just ask TestNG to run all the "front-end" tests, or "fast", "slow", "database" tests, etc.
6. Supports Dependent test methods, parallel testing, load testing, and partial failure.
7. Flexible plug-in API.
8. Support for multi threaded testing.

✍ **What is the difference between getWindowHandles() and getWindowHandle()?**

Ans: You can get the browser address using these commands. But if you use getWindowHandle(), you'll get the address of the current browser where the control is and

return type is a string. So, if you use getWindowHandles(), you will get the address of all the open browser and its return type is an iterator.

✍ **Why you have used TestNG in your framework?**

Ans:1. WebDriver has no native mechanism for generating reports.
2. TestNG can generate the report in a readable format
3. TestNG simplifies the way the tests are coded 4.
There is no more need for a static main method in our tests.
5. Uncaught exceptions are automatically handled by TestNG without terminating the test prematurely. These exceptions are reported as failed steps in the report.

✍ **What are different annotation present in TestNG?**

Ans: @BeforeSuite @AfterSuite
@BeforeClass @AfterClass
@BeforeTest @AfterTest
@BeforeGroups @AfterGroups
@BeforeMethod @AfterMethod
@DataProvider
@Test

✍ **What is priority feature in TestNG? In addition, how we can use this?**

Ans: 1. In TestNG "Priority" is used to schedule the test cases. When there are multiple test cases, we want to execute test cases in order.
2. In order to achieve, we use need to add annotation as @Test(priority=??). The default value will be zero for priority.
3. If we define priority as "priority=", these test cases will get executed only when all the test cases which don't have any priority as the default priority will be set to "priority=0".
@Test(priority=1), @Test(priority=2)

✍ **What is testng.xml file in TestNG?**

Ans: In testng.xml file we can specify multiple name (s) which needs to be executed. In a project there may be many classes, but we want to execute only the selected classes. We can pass class names of multiple packages also. If say suppose, we want to execute two classes in one package and other class from some other package. The below is the example testng.xml which will execute the class names that are specified.

✍ **How to group test cases in TestNG?**

Ans: TestNG allows us to perform sophisticated groupings of test methods. Using TestNG can we can execute only set of groups while excluding another set. This gives us the maximum flexibility in divide tests and doesn't require us to recompile anything if you want to run two different sets of tests back to back. Groups are specified in testng.xml file and can be used either under the or tag. Groups specified in the tag apply to all the tags underneath.

✍ **How to execute multiple test cases in Selenium?**

Ans: TestNG provides an option to execute multiple tests in a single configuration file (testng.xml). It allows to divide tests into different parts and group them in a single tests. We can group all the tests related to database into one group, Regression tests in one group. And all the test cases related to Unit test cases into one group and so on. In testng.xml file we can specify multiple name (s) which needs to be executed. In a project there may be many classes, but we want to execute only the selected classes. We can pass class names of multiple packages also. If say suppose, we want to execute two classes in one package and other class from some other package.

✍ **How to execute parallel test cases in Selenium?**

Ans: TestNG provides an ability to run test methods, test classes and tests in parallel. By using parallel execution, we can reduce the 'execution time' as tests are started and executed simultaneously in different threads. In testNG we can achieve parallel execution by two ways. One with testng.xml file and we can Configure an independent test method to run in multiple threads. First let us look at basic example for Parallel Execution of Test Methods using testng.xml. We will create a class with Two test methods and try to execute in different threads.

✍ **What is Data provider in TestNG?**

Ans-An important features provided by TestNG is the DataProvider feature. It helps you to write data-driven tests, which essentially means that same test method can be run multiple times with different data-sets. Please note that DataProvider is the second way of passing parameters to test methods (first way we already discussed in @Parameters example). It helps in providing complex parameters to the test methods as it is not possible to do this from XML. To use the DataProvider feature in your tests you have to declare a method annotated by @DataProvider and then use the said method in the test method using the ‘dataProvider‘ attribute in the Test annotation.

@DataProvider(name = "data-provider")

✍ **How to disable particular test case?**

Ans-Just add an attribute enabled=false in test declaration annotation.

Ex: @Test(enabled=false)

✍ **How to generate reports in TestNG?**

Ans- We just need to run an annotated TestNG annotation scripts and refresh the project you can see the test-output folder is generated in project explorer. Just click on to it, and then click on to the emailable-report.html you can view the testNG report in HTML format

✍ **Can you please explain what is apache maven?**

Ans- Apache Maven: Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

✍ **What is CI (Continuous integration) and what are different tools available in market.**

Ans- Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.

Jenkins GitHub Azure DevOps TeamCity Bitbucket Pipelines

✍ **How to integrate Jenkins with Selenium?**

Ans- 1. Open your web browser and then Navigate to Below URL http://jenkins-ci.org this is the official website of Jenkins.

2. Now download Jenkins.war file and save it.

3. Go to location where Jenkins.war is available.

4. Step 2- Open Command prompt knows as CMD and navigate till project home directory and Start Jenkins server Start- cmd> Project_home_Directory> java -jar jenkins.war

5. Open any browser and type the url http://localhost:8080

6. Click on > Manage Jenkins

7. Click on Configure System, Navigate to JDK section and Click on Add JDK button, Uncheck Install automatically check box so Jenkins will only take java which we have mention above.

8. Give the name as JAVA_HOME and Specify the JDK path

9. Part 3- Execute Selenium build using Jenkins

10.Part 4-Schedule your build in Jenkins for periodic execution

✍ **How to schedule test cases for nightly execution?**

Ans- Open Task Scheduler by clicking the Start button Picture of the Start button, clicking Control Panel, clicking System and Security, clicking Administrative Tools, and then double-clicking Task Scheduler. Administrator permission required If you're prompted for an administrator password or confirmation, type the password or provide confirmation.

✍ **How to read write excel files using Apache POI.**

Ans: Basics of APACHE POI There are two main prefixes which you will encounter when working with Apache POI:

 HSSF: denotes the API is for working with Excel 2003 and earlier.

 XSSF: denotes the API is for working with Excel 2007 and later. And to get started the Apache POI API, you just need to understand and use the following 4 interfaces: Workbook: high level representation of an Excel workbook. Concrete implementations are: HSSFWorkbook andXSSFWorkbook.

 Sheet: high level representation of an Excel worksheet. Typical implementing classes are HSSFSheet and XSSFSheet.

 Row: high level representation of a row in a spreadsheet. HSSFRow and XSSFRow are two concrete classes.

 Cell: high level representation of a cell in a row. HSSFCell and XSSFCell are the typical implementing classes.

✍ **Why and how will you use an Excel Sheet in your project?**

The reason we use Excel sheets is because it can be used as data source for tests. An excel sheet can also be used to store the data set while performing DataDriven Testing. These are the two main reasons for using Excel sheets.

When you use the excel sheet as data source, you can store the following:

 Application URL for all environments: You can specify the URL of the environment in which you want to do the testing like: development environment or testing environment or QA environment or staging environment or production/ pre-production environment.

 User name and password credentials of different environments: You can store the access credentials of the different applications/ environments in the excel sheet. You can store them in encoded format and whenever you want to use them, you can decode them instead of leaving it plain and unprotected.

 Test cases to be executed: You can list down the entire set of test cases in a column and in the next column, you can specify either Yes or No which indicates if you want that particular test case to be executed or ignored.

When you use the excel sheet for DataDriven Test, you can store the data for different iterations to be performed in the tests. For example while testing a web page, the different sets of input data that needs to be passed to the test box can be stored in the excel sheet.

✍ **What is Selendroid?**

Ans-Selendroid can be used to test already built apps. Those Android apps (apk file) must exist on the machine, where the selendroid-standalone server will be started. The reason for this is that a customized selendroid-server for the app under test (AUT) will be created. Both apps (selendroid-server and AUT) must be signed with the same certificate in order to install the apks on the device.

✍ **What is Appium?**

Ans- Appium aims to automate any mobile app from any language and any test framework, with full access to back-end APIs and DBs from test code. Write tests with your favorite dev tools using all the above programming languages, and probably more (with the Selenium WebDriver API and language-specific client libraries).

****

Cucumber Framework Interview Questions & Answers

1)What is Cucumber? Why is it used?

Cucumber is a testing tool based on Behavior Driven Development (BDD) framework. It is used to run functional tests written in plain text and develop test cases for software functionality. It plays a supporting role in automated testing. In other words, we can say that "Cucumber is a software tool used by the testers to develop test cases for the testing of behavior of the software." The main aim of the Behavior Driven Development framework is to make various project roles such as Business Analysts, Quality Assurance, Developers, etc., understand the application without diving deep into the technical aspects.

2) What language is used by Cucumber?

Gherkin is the language that is used by the Cucumber tool. It is a simple English representation of the application behavior. Gherkin language uses several keywords to describe the behavior of applications such as Feature, Scenario, Scenario Outline, Given, When, Then, etc.

3) What is meant by a feature file?

A feature file must provide a high-level description of an Application Under Test (AUT). The first line of the feature file must start with the keyword 'Feature' followed by the description of the application under test. A feature file may include multiple scenarios within the same file. A feature file has the extension .feature.

4) What are the various keywords that are used in Cucumber for writing a scenario?

Mentioned below are the keywords that are used for writing a scenario:

 Given When Then And

5) What is the purpose of a Scenario Outline in Cucumber?

Scenario outline is a way of parameterization of scenarios. This is ideally used when the same scenario needs to be executed for multiple sets of data, however, the test steps remain the same. Scenario Outline must be followed by the keyword 'Examples', which specify the set of values for each parameter.

6) What programming language is used by Cucumber?

Cucumber tool provides support for multiple programming languages such as Java, .Net, Ruby etc. It can also be integrated with multiple tools such as Selenium, Capybara, etc.

7) What is the purpose of the Step Definition file in Cucumber?

A step definition file in Cucumber is used to segregate the feature files from the underlying code. Each step of the feature file can be mapped to a corresponding method on the Step Definition file. While feature files are written in an easily understandable language like, Gherkin, Step Definition files are written in programming languages such as Java, .Net, Ruby, etc.

8) What are the major advantages of the Cucumber framework?

- Cucumber is an open-source tool.
- Plain Text representation makes it easier for non-technical users to understand the scenarios.

- It bridges the communication gap between various project stakeholders such as Business Analysts, Developers, and Quality Assurance personnel.
- Automation test cases developed using the Cucumber tool are easier to maintain and understand as well.
- Easy to integrate with other tools such as Selenium and Capybara.

9) Provide an example of a feature file using the Cucumber framework.

Following is an example of a feature file for the scenario 'Login into the application:
Feature: Login to the application under test. Scenario:
1. Login to the application.
2. Open the Chrome browser and launch the application.
3. When the user enters the username onto the UserName field.
4. And User enters the password into the Password field.
5. When the user clicks on the Login button.
6. Then validate if the user login is successful.

10) Provide an example of a Scenario Outline using the Cucumber framework.

The following is an example of a Scenario Outline keyword for the scenario 'Upload a file'. The number of parameter values to be included in the feature file is based on the tester's choice.
Scenario Outline: 1. Upload a file
2. Given that the user is on upload file screen.
3. When a user clicks on the Browse button.
4. And user enters <filename> onto the upload textbox.
5. And user clicks on the enter button.
6. Then verify that the file upload is successful.
Example:
|filename| |file1| |file2|

11) What is the purpose of the Behaviour Driven Development (BDD) methodology in the real world?

BDD is a methodology to understand the functionality of an applicatio the simple plain text representation. The main aim of the Behavior Driven Development framework is to make various project roles such as Business Analysts, Quality Assurance, Developers, and Support Teams understand the application without diving deep into the technical aspects.

12) What is the limit for the maximum number of scenarios that can be included in the feature file?

A feature file can contain a maximum of 10 scenarios, but the number can vary from project to project and from one organization to another. But it is generally advisable to limit the number of scenarios included in the feature file.

13) What is the use the of Background keyword in Cucumber?

Background keyword is used to group multiple given statements into a single group.
This is generally used when the same set of the given statement is repeated in each Scenario

of the feature file.

14) What symbol is used for parameterization in Cucumber?

Pipe symbol (|) is used to specify one or more parameter values in a feature file.

15) What is the purpose of Examples keyword in Cucumber?

Examples keyword is used to specify values for each parameter used in the scenario. Scenario Outline keyword must always be followed by the keyword Examples.

16) What is the file extension for a feature file?

File Extension for a feature file is .feature. A feature file is ideally written in a notepad file and is saved with the extension feature.

17) Provide an example of a step definition file in Cucumber.

Step definition corresponding to the step "Open Chrome browser and launch the application" may look like the code mentioned below:

```
@Given("^Open Chrome browser and launch the application$")
public void openBrowser()
{
driver = new ChromeDriver();
driver.manage().window().maximize();
driver.get("www.facebook.com");
}
```

18) What is the purpose of the Cucumber Options tag?

Cucumber Options tag is used to provide a link between the feature files and step definition files. Each step of the feature file is mapped to a corresponding method on the step definition file. Below is the syntax of Cucumber Options tag:

```
@CucumberOptions(features="Features",glue={"StepDefinition"})
```

19) How can Cucumber be integrated with Selenium WebDriver?

Cucumber can be integrated with the Selenium Webdriver by downloading the necessary JAR files.

Given below are the list of JAR files that are to be downloaded for using Cucumber with Selenium web driver:

- cucumber-core-1.2.2.jar
- cucumber-java-1.2.2.jar
- cucumber-junit-1.2.2.jar
- cucumber-jvm-deps-1.0.3.jar
- cucumber-reporting-0.1.0.jar
- gherkin-2.12.2.jar

20) When is Cucumber used in real-time?

Cucumber tool is generally used in real-time to write acceptance tests for an application. It is generally used by non-technical people such as Business Analysts, Functional Testers, etc.

21) What is the use of Behavior Driven Development in Agile methodology?

The advantages of Behavior Driven Development are best realized when non-technical users such as Business Analysts use BDD to draft requirements and provide the same to the developers for implementation.

In Agile methodology, user stories can be written in the format of feature file and the same can be taken up for implementation by the developers.

22) Explain the purpose of keywords that are used for writing a scenario in Cucumber.

- "Given" keyword is used to specify a precondition for the scenario.
- "When" keyword is used to specify an operation to be performed.
- "Then" keyword is used to specify the expected result of a performed action.
- "And" keyword is used to join one or more statements together into a single statement.

23) What is the name of the plugin that is used to integrate Eclipse with Cucumber?

Cucumber Natural Plugin is the plugin that is used to integrate Eclipse with Cucumber.

24) What is the meaning of the TestRunner class in Cucumber?

TestRunner class is used to provide the link between the feature file and the step definition file. The next question provides a sample representation of how the TestRunner class will look like. A TestRunner class is generally an empty class with no class definition.

25) Provide an example of the TestRunner class in Cucumber.

```
Package com.sample.TestRunner
importorg.junit.runner.RunWith;
importcucumber.api.CucumberOptions;
importcucumber.api.junit.Cucumber;
@RunWith(Cucumber.class)
@CucumberOptions(features="Features",glue={"StepDefinition"})
public class Runner
{       }
```

26) What is the starting point of execution for feature files?

When integrated with Selenium, the starting point of execution must be from the TestRunner class.

27) What do you understand by cucumber dry run?

Cucumber dry run is used to compile cucumber features files and step definitions. It is run to find any compilation errors. If it finds anyone, it will show when we use dry run.

28) Should any code be written within the TestRunner class?

No code should be written under the TestRunner class. It should include the tags @RunWith and @CucumberOptions.

29) What is the use of features property under the Cucumber Options tag?

Features property is used to let the Cucumber framework identify the location of the feature files.

30) What is the use of glue property under the Cucumber Options tag?

Glue property is used to let the Cucumber framework identify the location of step definition files.

31)What are the two build management tools that can be integrated with Cucumber?

Following are the two build management tools that can be integrated with Cucumber:

- Gradle
- Maven

32)What If You Don't Use The Cucumber Keywords In Test Steps?

Please note that it's not mandatory to write keywords in test steps.

For example, we can build a test step like the one shown in the next line.

e.g.- We are testing using cucumber.

33) What is the main aim of the Behavior Driven Development (BDD) framework?

The main aim of the Behavior Driven Development framework is to make various project roles such as Business Analysts, Quality Assurance, Developers, etc., understand the application without diving deep into the technical aspects. maximum number of steps that are to be written within a scenario are 3-4 steps.

34) What are the two files required to execute a Cucumber test scenario?

Following are the two files required to execute a Cucumber test scenario:

- Features
- Step Definition

35) What do you understand by a feature file?

A feature file is used to provide a high-level description of an Application Under Test (AUT). The first line of the feature file must start with the keyword 'Feature' followed by the description of the application under test. A feature file may include multiple scenarios within the same file, and the extension of the feature file must be ".feature."

37) What is the difference between Selenium and Cucumber?

Selenium and Cucumber are both open-source testing tools, and both are used for functional testing. But there are some differences between them.

Following are some critical differences between Selenium and Cucumber:

- Selenium is a web browser automation tool for web apps, while Cucumber is an automation tool for behavior-driven development that can be used with Selenium (or Appium).
- Selenium is used for automated UI testing, while Cucumber is used for acceptance testing.
- Selenium is preferred by technical teams (SDETs/programmers), while Cucumber is typically preferred by non-technical teams (business stakeholders and testers).
- Selenium can work independently of Cucumber. Cucumber depends on Selenium or Appium for step-definition implementation.
- In Selenium, the script creation is complex, while Cucumber is simpler than Selenium.

38) Why we have to use Cucumber with Selenium?

Cucumber and Selenium are both testing frameworks and prevalent technologies. Many organizations use Selenium for functional testing. Along with Selenium, these organizations integrate Cucumber with Selenium as Cucumber makes it easy to read and understand the application flow. The most significant benefit of using Cucumber with Selenium is that it

facilitates developers to write test cases in simple feature files easily understood by managers, non-technical stakeholders, and business analysts. It provides the facility to write tests in a human-readable language called Gherkin. The Selenium-Cucumber framework supports programming languages such as Java, .NET, PHP, Python, Perl, etc.

39) How can you use the Options tag in the Cucumber framework?

In the Cucumber framework, the Options tag is a part of the TestRunner file and comes in the form of an annotation called @CucumberOptions. It contains two parameters feature and glue.

 Feature parameter: The feature parameter is used to specify the path of the feature file.
 Glue parameter: The glue parameter is used to specify the path of the step definition file.

See the code implementation of TestRunner file with Option tag:

```
import org.junit.runner.RunWith;
import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;
@RunWith (Cucumber.class)
@CucumberOptions (
features = "src/test/java/features ",
glue = {"stepDefinitions"})
public class TestRunner {
}
```

We have to import org.junit.runner.RunWith for the @RunWith annotation and cucumber.api.CucumberOptions for the @CucumberOptions annotation.

40) What Are Cucumber Tags? And Why Do We Use Them?

Cucumber tags help in filtering the scenarios. We can tag the scenarios and then run them based on tags.

We can add tags to scenarios with <@> symbol.

We can use the following command to run a cucumber tagged scenario.

cucumber features -t @<tag_name>

Example: cucumber features -t @test

****

GIT and GITHUB Interview Questions & Answers

Q1) What is GIT?

ANS: Git is an open-source distributed version control system and source code management (SCM) system with an insistence to control small and large projects with speed and efficiency.

Q2) What is a repository in Git?

ANS: A repository consists of a list named .git, where git holds all of its metadata for the catalog. The content of the .git file is private to Git.

Q3) What are the advantages of using GIT?

- Data repetition and data replication is possible
- It is a much applicable service
- For one depository you can have only one directory of Git
- The network performance and disk application are excellent
- It is effortless to collaborate on any project
- You can work on any plan within the Git

Q4) Why do we require branching in GIT?

ANS: With the help of branching, you can keep your branch, and you can also jump between the different branches. You can go to your past work while at the same time keeping your recent work intact.

Q5) What is the purpose of 'git config'?

ANS: The 'Git config' is a great method to configure your choice for the Git installation. Using this command, you can describe the repository behavior, preferences, and user information.

Q6) What is a 'conflict' in git?

ANS: A 'conflict' appears when the commit that has to be combined has some change in one place, and the current act also has a change at the same place. Git will not be easy to predict which change should take precedence.

Q7) What is the difference between git pull and git fetch?

ANS: Git pull command pulls innovation or commits from a specific branch from your central repository and updates your object branch in your local repository. Git fetch is also used for the same objective, but it works in a slightly different method. When you behave a git fetch, it pulls all new commits from the desired branch and saves it in a new branch in your local repository. If you need to reflect these changes in your target branch, git fetch should be followed with a git merge. Your target branch will only be restored after combining the target branch and fetched branch.

To make it simple for you, remember the equation below:

Git pull = git fetch + git merge

Q8) How to resolve a conflict in Git?

ANS: If you need to resolve a conflict in Git, edit the list for fixing the different changes, and then you can run "git add" to add the resolved directory, and after that, you can run the 'git commit' for committing the repaired merge.

Q9) What is the purpose of the git clone?

ANS: The git clone command generates a copy of a current Git repository. To get the copy of a central repository, 'cloning' is the simplest way used by programmers.

Q10) What is git pull origin?

ANS: pull is a get and a consolidation. 'git pull origin master' brings submits from the master branch of the source remote (into the local origin/master branch), and then it combines origin/master into the branch you currently have looked out.

Q11) What does git commita?

ANS: Git commits "records changes to the storehouse" while git push " updates remote refs along with contained objects" So the first one is used in a network with your local repository, while the latter one is used to communicate with a remote repository.

Q12) Explain what is commit message?

ANS: Commit message is a component of git which shows up when you submit a change. Git gives you a content tool where you can enter the adjustments made to a commit.

Q13) What is the distinction between Git and Github?

Git	GitHub
Software	Service
Version control	Git repository hosting
Maintained by Linux	Maintained by Microsoft
Open-Source	Free or paid membership
No user management	Built-in user management
Locally installed	Hosted on the web
Minimal external tool configuration	Active marketplace for tool integration
Little to no competition	High competition

Q14) In Git, how would you return a commit that has just been pushed and made open?

ANS: There can be two answers to this question and ensure that you incorporate both because any of the below choices can be utilized relying upon the circumstance: Remove or fix the bad document in another commit and push it to the remote repository. This is a unique approach to correct a mistake. Once you have necessary changes to the record, commit it to the remote repository for that I will utilize.

git submit - m "commit message."

Make another commit that fixes all changes that were made in the terrible commit. to do this, I will utilize a command.

git revert <name of bad commit>

Q15) What does the committed item contain?

ANS: Commit item contains the following parts; you should specify all the three present below: A set of records, representing to the condition of a task at a given purpose of time

References to parent commit objects An SHAI name, a 40 character string that uniquely distinguishes the commit object.

Q16) Describing branching systems you have utilized?

ANS: Feature Branching: >A component branch model keeps the majority of the changes for a specific element within a branch. >At the point when the item is throughout tested and approved by automated tests, the branch is then converted into master.

Task Branching: >In this model, each assignment is actualized on its branch with the undertaking key included in the branch name. >It is anything but difficult to see which code actualizes which task, search for the task key in the branch name.

Release Branching: >Once the create branch has procured enough features for a discharge, you can clone that branch to frame a Release branch. >Making this branch begins the following discharge cycle so that no new features can be included after this point, just bug fixes, documentation age, and other release-oriented assignments ought to go in this branch. >When it is prepared to deliver, the release gets converged into master and labeled with a form number. >Likewise, it should be converged once again into creating a branch, which may have advanced since the release was started.

Q17) Explain the difference between `git merge` and `git rebase` and when you would use each?

ANS: 'git merge' merges changes from one branch into another, keeping distinguish branch histories. It forms merge commits that directly show where branches came together. 'git rebase' reforms history by using commits from one branch onto another, causing in a linear sequence of commits. Prefer 'merge' to keep distinct branch timelines and use 'rebase' for enhancing commit history before merging branches.

Q18) What language is used in GIT?

ANS: Git is mainly developed using the C programming language. The core features and commands of Git, containing its data structures and algorithms, are applied in C. This choice of language confirms productivity, speed, and portability across distinct operating systems and platforms.

Q19) How do you handle large files with Git?

ANS: To handle large files in Git, use Git LFS (Large File Storage). It tracks large files severally from your repository, storing them on a remote server. This prevents bloating your repository size and secures improved performance while operations like cloning and fetching.

Q20) Mention the various Git repository hosting functions.

ANS: Pikacode Visual Studio Online GitHub GitEnterprise SourceForge.net

****

CI/CD and Jenkins Interview Questions & Answers

Q #1) What is Jenkins?

Answer: Jenkins is a free open-source Continuous Integration tool and automation server to monitor continuous integration and delivery. It is written in Java. It is known as an automated Continuous Delivery tool that helps to build and test the software system with easy integration of changes to the system. Jenkins follows Groovy Scripting. Also, it enables developers to continuously check in their code and also analyze the post-build actions. The automation testers can use to run their tests as soon as the new code is added or code is modified.

Q #2) What are the features of Jenkins?

Answer: 1. Free open source. 2. Easy installation on various operating systems. 3. Build Pipeline Support. 4. Workflow Plugin. 5. Test harness built around JUnit. 6. Easy upgrades. 7. Rapid release cycle. 8. Easy configuration setup. 9. Extensible with the use of third-party plugins.

Q #3) What are the advantages of Jenkins? Why we use Jenkins?

Answer: Build failures are cached during the integration stage. Notifies the developers about build report status using LDAP (Lightweight Directory Access Protocol) mail server. Maven release project is automated with simple steps. Easy bug tracking. Automatic changes get updated in the build report with notification. Supports Continuous Integration in agile development and test-driven development.

Q #4) What is Continuous Integration in Jenkins?

Answer: Continuous integration is the process of continuously checking-in the developer's code into a version control system and triggering the build to check and identify bugs in the written code. In software development, multiple developers work on different software modules. While performing integration testing all the modules are being integrated together. It is considered as the development practice to integrate the code into the source repository Whenever the programmer/developer makes any change to the current code, then it automatically gets integrated with the system running on the tester's machine and makes the testing task easy and speedy for the system testers.

Continuous Integration comprises of: Development and Compilation Database Integration Unit Testing Production Deployment Code Labelling Functional Testing Generating and Analyzing Reports

Q #5) What is the difference between Hudson and Jenkins?

Answer: There is no difference between Hudson and Jenkins. Hudson was the former name of Jenkins, after going through several issues the name was changed to Jenkins.

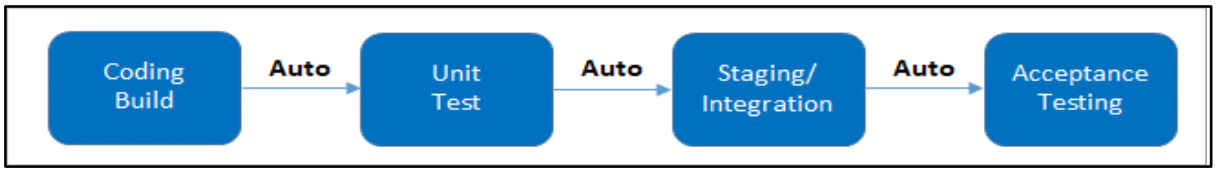
Q #6) Which command is used to start Jenkins?

Answer: You can follow the below-mentioned steps to start Jenkins: 1. Open Command Prompt 2. From the Command Prompt browse the directory where Jenkins. war resides 3. Run the command given below: D:\>Java –jar Jenkins.war

Q #7) What is the difference between Continuous Integration, Continuous Delivery, and Continuous Deployment?

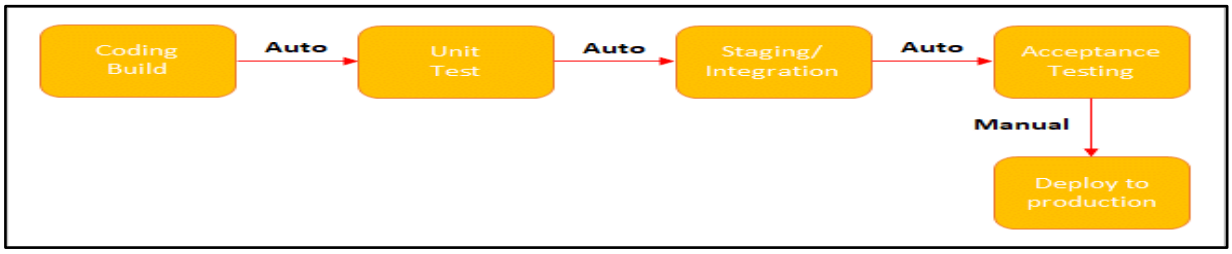
Answer: The diagrammatic representation given below can elaborate on the differences between Continuous Integration, Continuous Delivery, and Continuous Deployment more precisely.

Continuous Integration:



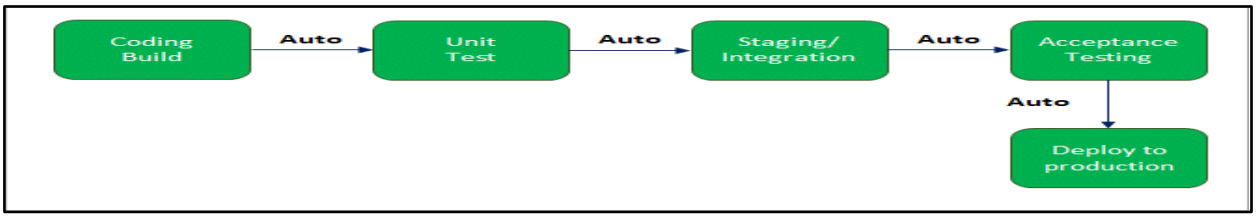
(It involves keeping the latest copy of the source code at a commonly shared hub where all the developers can check to fetch out the latest change in order to avoid conflict.)

Continuous Delivery:



(Manual Deployment to Production. It does not involve every change to be deployed.)

Continuous Deployment:



(Automated Deployment to Production. Involves every change to be deployed automatically.)

Q #8) What is Jenkins Pipeline? What is a CI CD pipeline?

Answer: The pipeline can be defined as the suite of plugins supporting the implementation and integration of continuous delivery pipelines in Jenkins.

Continuous integration or continuous delivery pipeline consists of build, deploy, test, release pipeline. The pipeline feature saves a lot of time and error in maintaining the builds. Basically, a pipeline is a group of build jobs that are chained and integrated in sequence.

Q #9) What is SCM? Which SCM tools are supported in Jenkins?

Answer: SCM stands for Source Control Management. SCM module specifies the source code location. The entry point to SCM is being specified as jenkins_jobs.scm. The job specified with 'scm' attribute accepts multiple numbers of SCM definitions.

Q #10) Which CI Tools are used in Jenkin?

Answer: Jenkins supported the following CI tools: 1. Jenkins 2. GitLab CI 3. Travis CI 4. CircleCI 5. Codeship 6. Go CD 7. TeamCity 8. Bamboo

Q #11) Which commands can be used to start Jenkins manually?

Answer: You can use the following commands to start Jenkins manually: 1. (Jenkins_url)/restart: To force restart without waiting for build completion. 2. (Jenkin_url)/safeRestart: Waits until all the build gets completed before restarting.

Q #12) How to make sure that your project build does not break in Jenkins?

Answer: You need to follow the below-mentioned steps to make sure that the Project build does not break: 1. Clean and successful installation of Jenkins on your local machine with all unit tests. 2. All code changes are reflected successfully. 3. Checking for repository synchronization to make sure that all the differences and changes related to config and other settings are saved in the repository.

Q #13) How will you define Post in Jenkins?

Answer: Post is a section that contains several additional steps that might execute after the completion of the pipeline. The execution of all the steps within the condition block depends upon the completion status of the pipeline. The condition block includes the following conditions – changed success, always, failure, unstable and aborted.

Q #14) What are Parameters in Jenkins?

Answer: Parameters are supported by the Agent section and are used to support various use-cases pipelines. Parameters are defined at the top-level of the pipeline or inside an individual stage directive.

Q #15) How can you secure Jenkins?

Answer: Securing Jenkins is a little lengthy process, and there are two aspects of securing Jenkins: (i) Access Control which includes authenticating users and giving them an appropriate set of permissions, which can be done in 2 ways. Security Realm determines a user or a group of users with their passwords. Authorization Strategy defines what should be accessible to which user. In this case, there might be different types of security based on the permissions granted to the user such as Quick and simple security with easy setup, Standard security setup, Apache front-end security, etc. (ii) Protecting Jenkins users from outside threats.

Q #16) How to create a backup and copy files in Jenkins?

Answer: In Jenkins, all the settings, build logs and configurations are stored in the JENKINS_HOME directory. Whenever you want to create a backup of your Jenkins you can back up JENKINS_HOME directory frequently. It consists of all the job configurations and slave node configurations. Hence, regularly copying this directory allows us to keep a backup of Jenkins. You can maintain a separate backfile and copy it whenever you need the same. If you want to copy the Jenkins job, then you can do so by simply replicating the job directory.

Q #17) How does Jenkins fit in with DevOps?

Answer: DevOps refers to software development practice that sort of blends & synchronizes the process of software development (Dev) with IT operations (Ops). Thus creating the whole development cycle feasible & shorter by constantly building fixes, builds, updates &

features. Jenkins plays an imperative role as it helps in this integration by automating the build, test as well as deployment process.

Q #18) How do you integrate GIT with Jenkins?

Answer: •Once you are at Jenkins Dashboard click on the "Manage Jenkins" button.

•Click on Manage Plugins

•On the Plugin page, select the GIT Plugin.

•Install the Git plugin and restart your Jenkins.

•Once you install the plugins, go to Manage Jenkins on your Jenkins dashboard. You will see your plugins listed among the rest.

Q #19) How Jenkins can be used for testing in different environments?

Answer: •Jenkins checks the Git repository at periodic intervals for any changes made in the source code.

•Each builds requires a different testing environment which is not possible for a single Jenkins server. To perform testing in different environments, Jenkins uses various Slaves as shown in the below diagram.

•Jenkins Master requests these Slaves to perform testing and generate test reports.

Q #20) Differentiate between Maven, Ant, and Jenkins?

Maven	Ant	Jenkins
Build automation and project management tool	Build automation tool	Continuous Integration/Continuous Delivery (CI/CD) tool
Dependency management, build lifecycle handling	Dependency management, build lifecycle handling	Dependency management, build lifecycle handling
Limited flexibility due to standard conventions	Limited flexibility due to standard conventions	Limited flexibility due to standard conventions
Supports CI/CD via Jenkins or other CI tools	Supports CI/CD via Jenkins or other CI tools	Supports CI/CD via Jenkins or other CI tools

****

"Job preparation is not just about resumes and interviews; it's about believing in yourself. You've got this!"

"Research, practice, and confidence—these are your keys to success. Prepare well, and the right job will find you."

Yours sincerely,

Krishna N.