**SELENIUM POM FRAMEWORK**

The following tools and frameworks are used in selenium pom framework.

Maven :- Maven is a build automation tool is hosted by Apache Software Foundation. It is a powerful project/build management tool based on concept of POM that includes project and configuaration information such as source directory, dependencies, plugins, Goals, test source directory.

Why we need to use Maven:-

1. Define project structure, dependencies, build and test management.
2. It automatically downloads the necessary files from the repository while building the project.
3. Configure the dependencies needed forbuilding testing and running the code.

TestNG Framework:- It is a automation testing and annotation based framework capable of making selenium tests easy to understand and of generating reports. So that you can easily conclude that how many tests are passed, failed and skipped. (NG stands for Next Generation).

Why we use TestNG :-

1. Multiple tests are grouped more easily by converting then to testing.xml.
2. Prioritisation of test cases.
3. Execution of multiple test cases on multiple browsers.
4. Can generate report in readable formate.
5. Can run same test multiple times without loops just by using key word called “Invocation Count”.

Listeners :- As the name suggest it “listen” to event defined in selenium script and behave accordingly. Used to implementing listeners interface. Allows us customizing the reports or logs.

ITestListeners :- An interface most used listener in TestNG with Selenium. The class in which we implement the listener overrides the ITestListener methods. It is used at test level.

ISuiteListeners:-An interface used at the suite level which has before and after suite methods i.e. OnStart and OnFinish methods.

Singleton Class :- one of the design pattern that restricts the instantiation of a class to one “single” instance.

Inheritance :- process in which one class acquires all properties and behavior of another/parent class. Allows code reusability. Java doesn’t support multiple inheritance to prevent ambiguity. So we have concept called Interface.

Ex:- public class HomePage extends BasePage{}

Implemented BasePage, has wrapper/helper methods and constructor. Every page which are used in testcases should inherit the BasePage. So that I can use those methods whenever needed.

Abstraction :- process of hiding the implementation details and showing only functionality to the user.

Encapsulation :- process of wrapping code and data together into a single unit.

Why:-

1. By providing only a setter or getter method, you can make the class **read only or write only**. In other words, you can skip the getter or setter methods.
2. provides you the **control over the data**.
3. to achieve **data hiding**

Interface :- An **interface in Java** is a blueprint of a class. It has static constants and abstract methods. The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction, loose coupling and multiple [inheritance in Java](https://www.javatpoint.com/inheritance-in-java). In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

Loggers:- An [API](https://www.javatpoint.com/api-full-form) that provides the ability to trace out the errors of the applications. When an application generates the logging call, the Logger records the event in the LogRecord. After that, it sends to the corresponding handlers or appenders. Before sending it to the console or file, the appenders format that log record by using the formatter or layouts.

Why:-

1. It provides the complete tracing information of the application.
2. It records the critical failure if any occur in an application.

Exceptions :- An exception (or exceptional event) is a problem that arises during the execution of a program. When an **Exception** occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.

We used try-catch blocks in page and test level to catch any exception and throwing the that exception through user defined exception i.e. MyException.

Locators :- you will find the location of webelements with the help of locators like by id, name, class, xpath, cssselector.

Ex:- By image = By.className(“classname”)

Lists:- in Java provides the facility to maintain the ordered collection. It contains the index-based methods to insert, update, delete and search the elements. It can have the duplicate elements also. We can also store the null elements in the list.

Maps:- Maps are perfectly for key-value association mapping such as dictionaries. Use Maps when you want to retrieve and update elements by keys, or perform lookups by keys.

Arrays:- Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

We used Array of maps to get the test data from xl file

POM :- a design pattern or a framework that we use in Selenium using which one can create an object repository of the different web elements across the application. To simplify, in the Page Object Model framework, we create a class file for each web page. This class file consists of different web elements present on the web page. Moreover, the test scripts then use these elements to perform different actions. Since each page’s web elements are in a separate class file, the code becomes easy to maintain and reduces code duplicity.

Uses:-

1. Makes code maintainable.
2. Makes code readable
3. Makes code reusable.

Waits :- The wait concept in Selenium overcomes this problem and gives a **delay between elements identification and actions performed on them**.

Need:- Most of the web applications are developed using [Ajax](https://www.guru99.com/php-ajax.html) and [Javascript](https://www.guru99.com/interactive-javascript-tutorials.html). When a page is loaded by the browser the elements which we want to interact with may load at different time intervals.

Not only it makes this difficult to identify the element but also if the element is not located it will throw an "**ElementNotVisibleException**" exception. Using Selenium Waits, we can resolve this problem.

Implicit wait :- 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 is 0. Once we set the time, the web driver will wait for the element for that time before throwing an exception. Selenium Web Driver has borrowed the idea of implicit waits from Watir.

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

Explicit wait :- 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. It gives better options than implicit wait as it waits for dynamically loaded Ajax elements.

Ex :- WebDriverWait wait = new WebDriverWait(WebDriverRefrence,TimeOut);

Wait.until(expectedcondition)

Fluent wait:- 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.

Ex:- Wait wait = new FluentWait(WebDriver reference)

.withTimeout(timeout, SECONDS)

.pollingEvery(timeout, SECONDS)

.ignoring(Exception.class);

Actions Class :- Action Class in Selenium is a **built-in 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.

Actions action = new Actions(WebDriver)

Action.moveToElement(WebElement).build()

Robot Class :- used to enable automated testing for implementations of Java platform. It generates input events in native systems for test automation, self-running demos and other applications where users need control over mouse and keyboard. Robot class is easy to implement and it can be easily integrated with an automated framework.

Need:- in certain Selenium automation tests, users need control over keyboard or mouse to interact with OS windows like download pop-ups, print pop-ups, etc. and native applications like notepad, calculator, etc. Selenium Webdriver cannot handle these pop-ups/applications, so inversion 1.3 of java, robot class was introduced which can handle OS pop-ups/applications.

Extent Reports :-  An open-source reporting library used in selenium test automation. It can produce more interactive reports, a dashboard view, graphical view, capture screenshots at every test step, and emailable reports. Extent reports are more customizable.