Ranjeet Kendre

Maven Project

Understanding the problem without Maven :

There are many problems that we face during the project development.

1) Adding set of Jars in each project: In case of, spring, hibernate frameworks,DDF,BDD we need to add set of jar files in each project.It must include all the dependencies of jars also.

2) Creating the right project structure: We must create the right project structure otherwise it will not be executed.

3) Building and Deploying the project: We must have to build and deploy the project so that it may work.

What is Maven?
Maven is a build automation tool .

What it does?
Maven simplifies the above mentioned problems. It does mainly following tasks.

It makes a project easy to build
It provides uniform build process (maven project can be shared by all the maven projects)
It provides project information (log document, cross referenced sources, mailing list, dependency list, unit test reports etc.)
It is easy to migrate for new features of Maven
Apache Maven helps to manage

Maven Repository
A maven repository is a directory of packaged JAR file with pom.xml file. Maven searches for dependencies in the repositories.
There are 3 types of maven repository:

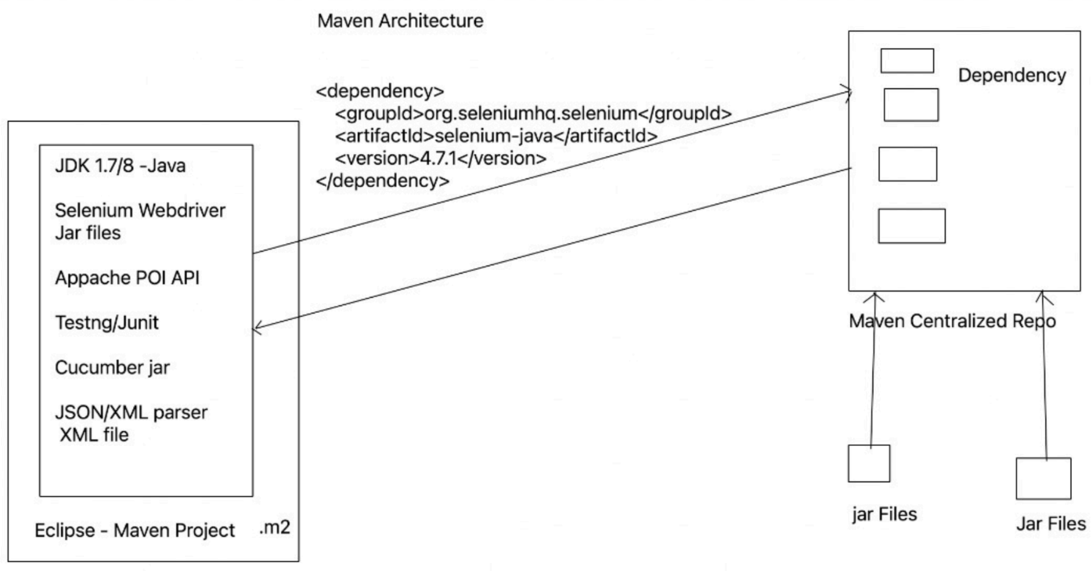
Central Repository
Maven searches for the dependencies in the following order:

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

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

What it does?
Maven simplifies the above mentioned problems. It does mainly following tasks.

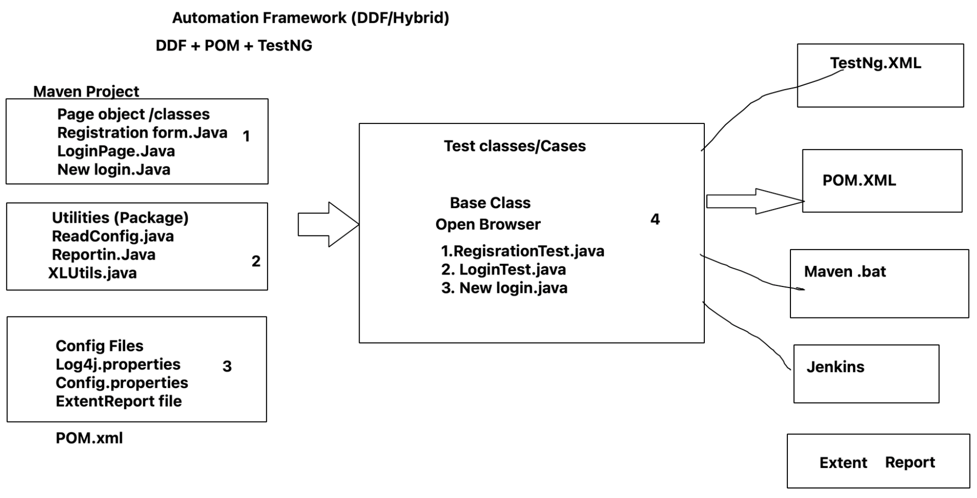
It makes a project easy to build
It provides uniform build process (maven project can be shared by all the maven projects)



It provides project information (log document, cross referenced sources, mailing list, dependency list, unit test reports etc.)
It is easy to migrate for new features of Maven
Apache Maven helps to manage

Builds
Documentation
Reporing
SCMs
Releases
Distribution
What is Build Tool
A build tool takes care of everything for building a process. It does following:

Generates source code (if auto-generated code is used)
Generates documentation from source code
Compiles source code
Packages compiled code into JAR of ZIP file
Installs the packaged code in local repository, server repository, or central repository



Framework Design Steps

Phase 1 - Implementation

1.Create maven project
2.Update pom.xml (Project object module)
3.Create pages object/classes
4.Create Test case
5.Create Base class
6.Add logs to test cases (log4j)
7.Read common values from properties file.
8.Run test case.
9. Add extent report (Extent report)
10. Create Data driver test cases
11. Adding new test cases

Phase 2 - Execution
1.   Run test cases through TestNg.xml file
2.   Run test cases through POM.xml file
3. Run test cases through Maven Command line interface(CMD?terminal)
4. Run test cases through .bat
5. Run test cases through Jenkins

Phase 3 - Maintenance
1.Creating repository in GITHub
2.Commit the project code in local repo.
3.Push the project code to GITHUB temp from ur local git repo

What is Logging?

Logging is a process that takes applications to a newer level with information logs on how the applications may or may not have performed/executed. It gives an exact idea of software performance, including any shortcomings.
Log4j in Selenium is one such logging framework that helps in gathering information in the form of logs or log files

Why Use Log4j in Selenium?

Use Log4j logging framework in Selenium for the following reasons:

- Log4j logging framework can help in debugging applications easily. With different log levels, it becomes easier to sort information based on categories.
- The logging framework is open source and can help in logging in different log levels and suppress logs in different environments – which ultimately improves application performance.
- The framework produces better output in general.
- With three components and clarity in usage, it becomes easier to use and configure log4j in Selenium.

Components of Log4j

The Log4j logging framework comprises the following components:

- Logger
- Appenders
- Layout

Logger

The function of the logger in Log4j is basically storing and capturing all the necessary logging information that will be generated using the framework.
To truly understand its functioning, let's dig a little deeper and discuss the logger class, and log level methods. The loggers also decide which priority is going to be captured.

- **Logger Class** – To fully use the logger, create an instance for a logger class where all the generic methods will be at the user's disposal, required to use Log4j.
- **INFO –** This level will log the progress of the application.

Appenders

The appender basically grabs information from the logger and writes log messages to a file or any other storage location. The following are some of the appenders one can use for Log4j:

- **FileAppender –** This will append the log messages to a file.
- **RollingFileAppender –** It will perform the same function as FileAppender, but users will be able to specify the maximum file size. Once the limit is exceeded, the appender will create another file to write the messages.

- **DailyRollingFileAppender –** It specifies the frequency by which the messages are to be written to the file.
- **ConsoleAppender –** In this, the appender will simply write the log messages in the console.

Layout
The layout is where the format in which log messages will appear is decided. There are several layouts one can use for log messages:

- **Pattern Layout –** The user must specify a conversion pattern based on which the logs will be displayed. Otherwise, it takes the default conversion pattern in case of "no pattern specified".
- **HTML Layout –** In this layout, the format of logs will be in the form of an HTML table.
- **XML Layout –** This will show the logs in an XML format.

what are the three principal components of Log4j?

The three principal components of Log4j are

- loggers: Responsible for capturing logging information.
- appenders: Responsible for publishing logging information to various preferred destinations.
- layouts: Responsible for formatting logging information in different styles.

What are the logging methods provided by logger class?

Logger class provides a variety of methods to handle logging activities. To obtain a logger object it provides two static methods

- Public static logger getRootLogger();
- Public static logger getLogger(String name);

Log4j.configuration

```
// Here we have defined root logger
log4j.rootLogger=INFO,CONSOLE,R,HTML,TTCC

// Here we define the appender
log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
log4j.appender.R=org.apache.log4j.RollingFileAppender
log4j.appender.TTCC=org.apache.log4j.RollingFileAppender
log4j.appender.HTML=org.apache.log4j.FileAppender

// Here we define log file location
log4j.appender.R.File=./Logs/ExecutionLog.log
```

```
log4j.appender.TTCC.File=./Logs/ExecutionLog1.log
log4j.appender.HTML.File=./Logs/ExecutionLog.html

// Here we define the layout and pattern
log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
log4j.appender.CONSOLE.layout.ConversionPattern= %5p [%t] (%F:%L)-
%m%n
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d - %c -%p - %m%n
log4j.appender.TTCC.layout=org.apache.log4j.TTCCLayout
log4j.appender.TTCC.layout.DateFormat=ISO8601
log4j.appender.HTML.layout=org.apache.log4j.HTMLLayout
log4j.appender.HTML.layout.Title=Application log
log4j.appender.HTML.layout.LocationInfo=true
```

Configuration .properties

What is the use of config.properties file in selenium

- Properties are used to externalize the data which is configurable and if you put that data in your code (test script) you have to build the code each time you want to change the value of the property. The main advantage of properties is that they are outside your source code and you can change them anytime.
- Each parameter is stored as a pair of strings, one storing the name of the parameter (called the *key*), and the other storing the value.
- Whatever constant and generic entities are there in our project, we can define in config.properties file

How to create config.properties file:

- Open eclipse. Right click on the project, select New→ file→ give file name as "config.properties"→ Finish. We cannot write java code in this. It is simple text file. Make sure to use extension as "properties" and not "property".
- Write the following content in properties file in key: value pair format:

Example :

```
BaseURL=https://demo.nopcommerce.com/login
Username=Ketan11@gmail.com
Password=Ketan@123
Chromdriver=/Users/ranjeetkendre/Documents/chromedriver
Firefox=/Users/ranjeetkendre/Documents/geckdriver
```

How to read config.properties file:

- Create a java class named ReadPropertyFile.java inside the same project where config.properties file is created → include “public static main” method and Finish.
- First we have to create object of Properties class.

Properties prop=new Properties(); *// This class is available in java*

3. Create object of **FileInputStream** and give property file location as fileInputStream parameter (which property file is to be read)

FileInputStream ip= new FileInputStream(“location of property file”);

4. Now we have to load the property file. Use properties object to load property file prop.load(“fileInputStream object”)

5. Now, once the config file is loaded, we need to read the properties of config file. Properties object gives us a .getProperty method which takes the key of the property as a parameter and return the value of the matched key from the .properties file.
System.out.println(prop.getProperty(“name”));

Make sure you use the same key as mentioned in properties file. This is case sensitive.

Note - Now in future, if your browser is firefox, you just need to update your properties file, instead of editing the class files/ test-scripts. Avoid to make changes in script. Nothing should be hard-coded at test-script level.

Final code should look something like this

package Com.Ecommerce.Configuration;

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Properties;

public class Readconfiguration {

	Properties prop;
	public Readconfiguration() {

		File src = new File(
				"/Users/ranjeetkendre/eclipse-workspace/
28_Oct_Automation_Framework2/src/test/resources/Configuration/
Config.properties");

		prop = new Properties();

		try {
			FileInputStream fis = new FileInputStream(src);
			try {
				prop.load(fis);
			} catch (IOException e) {
				System.out.println(e.getMessage());
				e.printStackTrace();
```

```
                }
            } catch (FileNotFoundException e) {
                System.out.println(e.getMessage());
                e.printStackTrace();
            }
        }
        public String getApllicationURL() {
            String URL = prop.getProperty("BaseURL");
            return URL;
        }
        public String Username() {
            String Username = prop.getProperty("Username");
            return Username;
        }
        public String Password() {
            String Passsowrd = prop.getProperty("Password");
            return Passsowrd;
        }
}
```

Implemented Retry Failed Tests in TestNG

You must have seen random failure during an automated test run. These failures might not necessarily be because of product bugs. These failure can be because of following reasons

- Random browser issues or browser becoming unresponsive
- Random machine issues
- Server issues like unexpected delay in the response from server •

These failure are genuine and should be investigated but these failures are not necessarily because of a bug in the product. TestNG provides a wonderful feature using which you can retry a test case multiple times before declaring it as Failed. What it means is that, if you see a failure can just automatically rerun the test to make sure that test is consistently failing. This way it reduces false failures because of random issues and you spend more time debugging true failures.
In order achieve this we have to first understand the org.testng.IRetryAnalyzer interface. The interface definition is
This interface has only one method public boolean retry(ITestResult result);
This method will be called once a test method fails
Implement IRetryAnalyzer to Retry Failed Test in TestNG Framework
Specifying retryAnalyzer during runtime
In this case you would need to
implement *ITestAnnotationTransformer* interface. *ITestAnnotationTransformer* interface falls under a broad category of interfaces called TestNG Listeners. You can read

How to Implement IRetryAnalyzer to Retry Failed Test in TestNG Framework

in a most design conscious way, so that we can have maintainable and easy to manage tests. This post will require you to have a good understanding annotation. We will create a custom annotation which we will use on a *@Test* Method so that we can specify the retry count directly in the Test

```
public class Mytransfomer implements IAnnotationTransformer{

       public void transform(ITestAnnotation annotation,
Class testClass, Constructor testConstructor, Method
testMethod) {

        annotation.setRetryAnalyzer(RetryAnalyzer.class);

            }
}

package Com.Ecommerce.Utilities;
public class RetryAnalyzer implements IRetryAnalyzer{

    int count =0;

    int RetrLimit= 0;  //2<2

    public boolean retry(ITestResult result) {

        if(count<RetrLimit) {
            count++;
            return true;
        }
        return false;
    }

}
```

What are Extent Reports?

Extent Reports is an open-source reporting library useful for test automation. It can be easily integrated with major testing frameworks like TestNG, JUNIT etc. These reports are HTML documents that depict results as pie charts. They also allow the generation of custom logs, snapshots, and other customized details.
Once an automated test script runs successfully, testers need to generate a test execution report. While TestNG does provide a default report, they do not provide the details.

TestNG Listeners

TestNG provides the @Listeners annotation which listens to every event that occurs in a selenium code. Listeners are activated either before the test or after the test case. It is an interface that modifies the TestNG behavior. For example, when you are running a test case through selenium test case fails. We need a screenshot of the test case that has been failed, to achieve such scenario, TestNG provides a mechanism, i.e., Listeners. When the test case failure occurs, then it is redirected to the new block written for the screenshot.

Listeners are implemented by the **ITestListener** interface. An ITestListener interface has the following methods:

ITestListener has following methods

- **OnStart-** OnStart method is called when any Test starts.
- **onTestSuccess-** onTestSuccess method is called on the success of any Test.
- **onTestFailure-** onTestFailure method is called on the failure of any Test.
- **onTestSkipped-** onTestSkipped method is called on skipped of any Test.
- **onTestFailedButWithinSuccessPercentage-** method is called each time Test fails but is within success percentage.
- **onFinish-** onFinish method is called after all Tests are executed.

How to create the TestNG Listeners

We can create the TestNG Listeners in two ways. First we can use the @Listeners annotation within the class and second way to use the within the suite

Benefits of using Extent Reports

- it can be integrated with TestNG and JUnit
- If required, screenshots can be captured and displayed for each step in a test
- They allow testers to track multiple test case runs in a single test suite
- They show the time needed for test execution
- They can be customized to graphically represent each step in a test.

```
<!-- Plugins required for execution test cases-->
    <build>
        <resources>
            <resource>
                <directory>src/main/java/resources</
directory>
                <filtering>true</filtering>
            </resource>
        </resources>
        <plugins>

            <plugin>
                <groupId>org.apache.maven.plugins</
groupId>
                <artifactId>maven-surefire-plugin</
artifactId>
                <version>3.1.2</version>
                <configuration>
                    <suiteXmlFiles>
                        <suiteXmlFile>testng.xml</
suiteXmlFile>
                    </suiteXmlFiles>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</
groupId>
                <artifactId>maven-compiler-plugin</
artifactId>
                <version>3.8.0</version>
                <configuration>
                    <source>1.7</source>
                    <target>1.7</target>
                </configuration>
            </plugin>

        </plugins>
    </build>
```

Execution :

1.Run test cases through Testing.xml
2.Run test cases through Pom.xml
Dependencies - download jar
Plugins - to run the test cases
Add
maven-surefire-plugin
maven-compiler-plugin

Pom.xml - Run as maven test

3.Run test cases through CMD/Terminal

You need to install maven on window
Maven download link
https://maven.apache.org/download.cgi

1. Add MAVEN_HOME in environment variable
Right click on MyComputer -> properties -> Advanced System Settings -> Environment variables -> click new button

2. Add Maven Path in environment variable

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

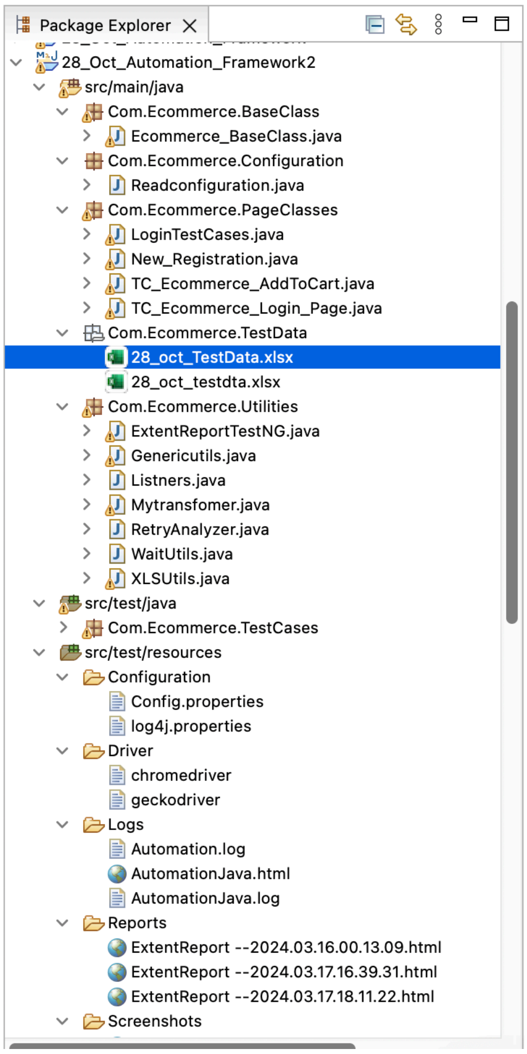
4.Run test cases through Jenkins pipeline

How To Explain Test Automation Framework To The Interviewer

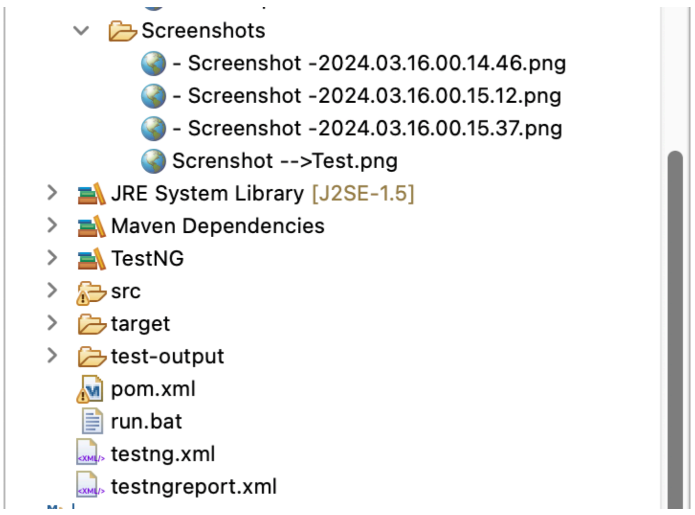
We need to specify in and out of our Test Automation Framework such as programming **language** used, **Type of framework** used, Test Base Class (Initializing WebDriver, Implicit Waits), How we separate Element locators and tests (Page Objects, Page Factory), Utility functions file, Property files, TestNG annotations, How we parameterize tests using Excel files, How we capture error screenshots, Generating reports(Extent Reports), Emailing reports, Version Control System used and Continues Integration Tool used.

Language: In our Selenium Project we are using Java language. Even though Selenium supports multiple languages, we are using Java language is just because most of the automation developers have knowledge on Selenium with Java.

Framework Structure



```
Package Explorer
28_Oct_Automation_Framework2
  src/main/java
    Com.Ecommerce.BaseClass
      Ecommerce_BaseClass.java
    Com.Ecommerce.Configuration
      Readconfiguration.java
    Com.Ecommerce.PageClasses
      LoginTestCases.java
      New_Registration.java
      TC_Ecommerce_AddToCart.java
      TC_Ecommerce_Login_Page.java
    Com.Ecommerce.TestData
      28_oct_TestData.xlsx
      28_oct_testdta.xlsx
    Com.Ecommerce.Utilities
      ExtentReportTestNG.java
      Genericutils.java
      Listners.java
      Mytransfomer.java
      RetryAnalyzer.java
      WaitUtils.java
      XLSUtils.java
  src/test/java
    Com.Ecommerce.TestCases
  src/test/resources
    Configuration
      Config.properties
      log4j.properties
    Driver
      chromedriver
      geckodriver
    Logs
      Automation.log
      AutomationJava.html
      AutomationJava.log
    Reports
      ExtentReport --2024.03.16.00.13.09.html
      ExtentReport --2024.03.17.16.39.31.html
      ExtentReport --2024.03.17.18.11.22.html
    Screenshots
```



```
    Screenshots
      - Screenshot -2024.03.16.00.14.46.png
      - Screenshot -2024.03.16.00.15.12.png
      - Screenshot -2024.03.16.00.15.37.png
      Screnshot -->Test.png
  JRE System Library [J2SE-1.5]
  Maven Dependencies
  TestNG
  src
  target
  test-output
  pom.xml
  run.bat
  testng.xml
  testngreport.xml
```

POM: As per the Page Object Model, we have maintained a class for every web page. Each web page has a separate class and that class holds the functionality and members of that web page. Separate classes for every individual test.

Packages: We have separate packages for *Pages* and *Tests*. All the *web page* related classes come under the **Pages** package and all the *tests* related classes come under **Tests** package.
For example, *Home Page* and *Login Page* have separate classes to store element locators. For the *login test,* there would be a separate class which calls the methods from the *Home Page* class and *Login Page* class.

Test Base Class: Test Base class (TestBase.java) deals with all the common functions used by all the pages. This class is responsible for loading the configurations from properties files, Initializing the WebDriver, Implicit Waits, Extent Reports, and also to create the object of FileInputStream which is responsible for *pointing towards the file from which the data should be read.*

Utility Class : Utility class (TestUtil.java) stores and handles the functions (The code which is repetitive in nature such as waits, actions, capturing screenshots, accessing excels, sending email, etc.,) which can be commonly used across the entire framework. The reason behind creating a utility class is to achieve reusability. This class extends the TestBase class to inherit the properties of TestBase in TestUtil.

Properties file: This file (***config.properties***) stores the information that remains static throughout the framework such as browser-specific information, application URL, screenshots path, etc.
All the details which change as per the environment and authorization such as URL, Login Credentials are kept in the *config.properties* file. Keeping these details in a separate file makes it easy to maintain.

Screenshots: Screenshots will be captured and stored in a separate folder and also the screenshots of failed test cases will be added to the extent reports.

Test Data: All the historical test data will be kept in an excel sheet (*controller.xlsx*). By using *'controller.xlsx'*, we pass test data and handle data-driven testing. We use Apache POI to handle excel sheets.

TestNG: Using TestNG for Assertions, Grouping, and Parallel execution.

Log4j : Log4j logging framework can help in debugging applications easily. With different log levels, it becomes easier to sort information based on categories.

Extent Reports: For the reporting purpose, we are using Extent Reports. It generates beautiful HTML reports. We use the extent reports for maintaining logs and also to include the screenshots of failed test cases in the Extent Report.

Maven
Using Maven for build, execution, and dependency purpose. Integrating the TestNG dependency in the POM.xml file and running this POM.xml file using Jenkins.

Version Control Tool: We use Git as a repository to store our test scripts.

Jenkins: By using Jenkins CI (Continuous Integration) Tool, we execute test cases on a daily basis and also for nightly execution based on the schedule. Test Results will be sent to the peers using Jenkins.

List of Challenges faced in project.

1) Domain knowledge of application
2) vast application, flows, functionalities & module integration was
3) use of dynamic Xpath
4) Handling multiple exceptions while test execution. Eg Stale Element Exception, NullPointer Exception
5) Dependencies of test Cases on one another
6) Execution of simple text Case work but files in Suite
8) maintenance of test Data & pre conditions on multiple
9) multiple unexpected. Pop up handling.
10) Application freeze because of multiple user using single part.

1. What is a Framework?

A framework defines a set of rules or best practices which we can follow in a systematic way to achieve the desired results.

2. Tell me some popular Test Automation Frameworks?
There are different types of test automation frameworks and the most common ones are:

- Data Driven Testing Framework
- Keyword Driven Testing Framework
- Hybrid Testing Framework
- Behavior Driven Development Framework

3. Why Framework?

In a test automation project, we do perform different tasks by using different types of files. To organize and manage all the files and to finish all the tasks in a systematic approach we use a framework.

4. Have you created any Framework?
If you are a beginner: *No, I didn't get a chance to create a framework. I have used the framework which is already available.*
If you are an experienced tester: *Yes, I have created a framework (Or) No, but I have involved in the creation of the framework.*

5. What are the advantages of using Test Automation Framework?
1. Saves time and money. Automation testing is faster in execution
2. Reusability of code. Create one time and execute multiple times with less or no maintenance
3. Easy reporting. It generates automatic reports after test execution

4. Easy for compatibility testing. It enables parallel execution in combination of different OS and browser environments
5. Low cost maintenance. It is cheaper compared to manual testing in a long run
6. Automated testing is more reliable
7. Automated testing is more powerful and versatile
8. It is mostly used for regression testing. Supports execution of repeated test cases
9. Minimal manual intervention. Test scripts can be run unattended
10. Maximum coverage. It helps to increase the test coverage

6. Which Test Automation Framework you are using and why?
Some of the Test Automation Frameworks are:

- Data Driven Testing Framework
- Keyword Driven Testing Framework
- Hybrid Testing Framework

7. Mention the name of the framework which 'you are currently using' or which 'you have hands on experience'.

Example:
Answers should be, *Already the organization which I am working for is using that particular framework* or *I have an experience on that particular framework* or *It's easy to handle all my scripts to execute and generate logs, screenshots and reports* by using this framework.

8. Can you explain the Framework which you have used in your Selenium Project?

9. Where you have applied OOPs in your Automation Framework?

Check this link for detailed answer

10. What is Automation testing? What are the advantages of Automation Testing?

Automation testing is the process of testing the software using an automation tool to find the defects. In this process, executing the test scripts and generating the results are performed automatically by automation tools. Some most popular tools to do automation testing are HP QTP/UFT, Selenium WebDriver, etc.,
For advantages refer to question 5 of this post "Test Automation Framework Interview Questions"

11. What are the most popular testing tools for functional testing?

1. Selenium
2. QTP(Quick Test Professional) / UFT(Unified Functional Testing)

12. Why do you prefer Selenium Automation Tool?

1. Free and open source
2. Have large user base and helping communities
3. Cross-browser compatibility
4. Platform compatibility
5. Multiple programming languages support

13. What type of test cases do you pick up to automate?

I focus on the test cases which should be executed in a repetitive manner such as regression test cases, smoke and sanity test cases

14. What type of test cases you won't pick up to automate?
Before picking up the test cases to automate, I do check whether the application is stable or not. So based on this, I don't pickup test cases when the AUT changes frequently and the test cases which I run rarely and run only one time. When I do usability and exploratory testing.

15. How many test cases you have automated per day?
It depends on Test case scenario complexity and length. I did automate 2-5 test scenarios per day when the complexity is limited. Sometimes just 1 or fewer test scenarios in a day when the complexity is high.

Framework Explanation :

We primarily use Java as our programming language and TestNG as the testing framework. Maven is employed for managing all the dependencies required for our project, while GitHub serves as our version control system for checking in our code files.

Our framework is constructed using a data-driven approach in conjunction with the Page Object Model. Test data is sourced from external files to ensure we avoid hard-coding it. Apache POI is utilized to extract test data from Excel files, and static information such as browser specifics, application URLs, usernames, and passwords are stored in config.properties.

For detailed reporting, we depend on Extent Reports as it plays a critical role in our workflow. Additionally, logging is implemented to capture all the details of the automation workflow for debugging purposes.

In the event of failed test cases, we capture screenshots and attach them to the automation report. Our file structure is designed to be highly modular, with page classes created for all common components, not restricted to specific pages. Whenever a component is shared across different pages, we create page objects for it, ensuring different classes can utilize them.
We strictly adhere to the DRY (Don't Repeat Yourself) principle to maintain efficient and non-redundant code.

TestNG XML files are used to manage different test suites such as Sanity and Regression Test Suites. Jenkins CI/CD pipelines are leveraged to execute these test cases across various test suites.