Class :21

Agenda

Automation Framework Concepts

Automation Testing Process

Java OOPS to build Automation Framework

Top Ten Test Automation Frameworks

Data Driven Framework in Selenium

Reading the Data from Excel Sheet

What is Automation Framework?

- A Framework defines a set of rules or best practices which user can follow in a systematic way to achieve the desired results.
- In a test automation project, user 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 user use a framework.
- Automation Framework is not a single tool or process, but it is a collection of tools and processes working together to support automated testing of any application. It integrates various functions like libraries, test data, and various reusable modules.
- Without a framework, a single test case may have a million lines of code and it will test the entire functionality of the application. Now, suppose user need to modify any functionality later time so is it possible to read the entire code again for a particular functionality?

  Answer will be No, because reading so huge test case will be tough and boring, and you also need lots of time to analyze the entire code.

Automation Framework Concepts

There are different types of test automation frameworks, and the most common ones are:

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

- Behavior Driven Development Framework

Advantages of using Test Automation Framework:

Saves time and maintain Schedule: Automation testing is faster in execution and definitely help to maintain overall project timeline

Reusability of code: Create one time and execute multiple times with less or no maintenance

Easy reporting: It generates automatic reports after test execution

Easy for compatibility testing: It enables parallel execution in combination of different OS and browser environments

Low cost maintenance: It is cheaper compared to manual testing in a long run

Reliability: Automated testing is more reliable as less manual intervention is required Repetitive multiple execution for same set of test cases: It is mostly used for regression testing, supports execution of repeated test cases

Minimal manual intervention: Test scripts can be run unattended Maximum coverage. It helps to increase overall test coverage

Automation Framework Concepts

Below are some of the key parameters that a software tester needs to keep in mind while developing any test automation framework.

Handle scripts and data separately:

Automated test scripts should be clearly separated from the input data store (e.g. XML, Ms-Excel files, Flat files or Databases), so that no modifications are required to the test scripts whenever data has to be changed for multiple input values.

Library:

A library should contain all reusable components and external connections such as databases, generic functions, application functions etc. Software testers should be exposed only to the implemented libraries and tests should be performed by invoking these libraries.

Coding Standards: Scripting standards should always be maintained across the test automation framework, which will discourage individual coding practices and help in maintaining code uniformity, which makes it easier for software testers and developers to interpret.

Extensibility and Maintenance:

An ideal test automation framework should steadily support all new enhancements to the software application and allow modification of existing features e.g. A reusable library can be created, which would help in enhancing application features with minimal effort.

Script/Framework Versioning:

Versions of framework / scripts should be maintained either in a local repository or versioning tool, which would help in easy monitoring of changes to the software code.

Types of Automated Testing

Smoke Testing

Unit Testing

Integration Testing

Functional Testing

Keyword Testing

Regression Testing

Data Driven Testing

Black Box Testing

Automation Testing Process

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

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

What type of test cases you won't pick up to automate?

Before picking up the test cases to automate, user should check whether the application is stable or not. So based on this, they don't pickup test cases when the AUT changes frequently and the test cases which they run rarely and run only one time.

Automated Testing Process:

Following common steps are followed in most of the Automation Testing Project

Step 1) Test Tool Selection

Step 2) Define scope of Automation

Step 3) Planning, Design and Development

Step 4) Test Execution

Step 5) Maintenance

Automation Testing Process

Test tool selection:

Test Tool selection largely depends on the technology the Application Under Test (AUT) is built on. For instance, QTP does not support Informatica. So QTP cannot be used for testing Informatica applications. It's a good idea to conduct a Proof of **Concept of Tool on AUT.**

Define the scope of Automation:

The scope of automation is the area of user Application Under Test which will beautomated. Following points help determine scope:

- The features that are important for the business
- Scenarios which have a large amount of data
- Common functionalities across applications
- Technical feasibility
- The extent to which business components are reused
- The complexity of test cases
- Ability to use the same test cases for cross-browser testing

Automation Testing Process

Planning, Design and Development:

During this phase, user create an Automation strategy & plan, which contains the following details:

- Automation tools selected
- Framework design and its features
- In-Scope and Out-of-scope items of automation
- Automation testbed preparation
- Schedule and Timeline of scripting and execution
- Deliverables of Automation Testing

Test Execution:

Automation Scripts are executed during this phase. The scripts need input test data before there are set to run. Once executed they provide detailed test reports. Execution can be performed using the automation tool directly or through the Test Management tool which will invoke the automation tool.

Example: Quality center is the Test Management tool which in turn it will invoke UFT for execution of automation scripts. Scripts can be executed in a single machine or a group of machines. The execution can be done during the night, to save time.

Test Automation Maintenance Approach:

Test Automation Maintenance Approach is an automation testing phase carried out to test whether the new functionalities added to the software are working fine or not. Maintenance in automation testing is executed when new automation scripts are added and need to be reviewed and maintained in order to improve the effectiveness of automation scripts with each successive release cycle.

Java OOPS to build Automation Framework

Important of four Java OOPs concepts and the ways how it is used in Selenium Automation Framework:

1) DATAABSTRACTION

- Data Abstraction is the property by virtue of which only the essential details are displayed to the user. The trivial or the non-essentials units are not displayed to the user. In java, abstraction is achieved by interfaces and abstract classes. User can achieve 100% abstraction using interfaces. In Selenium, WebDriver itself acts as an interface. Consider **the below statement:**

  WebDriver driver = new ChromeDriver();

- User initialize the Chrome Browser using Selenium Webdriver. It means they are creating a reference variable (driver) of the interface (WebDriver) and creating an Object. Here WebDriver is an Interface and ChromeDriver is a class.
- User can apply Data Abstraction in a Selenium framework by using the Page Object Model design pattern. They can define all locators and their methods in the page class. User can use these locators in their tests but they cannot see the implementation of their underlying methods. So user only show the locators in the tests but hide the implementation. This is a simple example of how user can use Data Abstraction in Automation Framework.

2) ENCAPSULATION

Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. Encapsulation can be achieved by declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables.

All the classes in an Automation Framework are an example of Encapsulation. In Page Object Model classes, user declare the data members using @FindBy and initialization of data members will be done using Constructor to utilize those in methods.

3) INHERITANCE

Inheritance is the mechanism in java by which one class (Child or Sub Class) is allow to inherit the features (fields and methods) of another class (Parent or Base Class).

User can apply Inheritance in Automation Framework by creating a Base Class to initialize the WebDriver interface, browsers, waits, reports, logging,etc. and then they can extend this Base Class and itsmethods in other classes like Tests or Utilities. This is a simple example of how user can apply Inheritance in our framework.

4) POLYMORPHISM

The word polymorphism means having many forms. Polymorphism allows user to perform a single action in different ways. In Java polymorphism can be achieved by two ways:

Method Overloading: When there are multiple methods with same name but different parameters then these methods are said to be overloaded. Methods can be overloaded by varying number of arguments or/and changing type of arguments. In Selenium Automation, Implicit wait is an example of Method Overloading. In Implicit wait user use different time stamps such as SECONDS, MINUTES, HOURS etc.

Method Overriding: It occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be overridden. In Selenium Automation, Method Overriding can be achieved by overriding any WebDriver method. In the WebDriver interface, user use two different methods for navigating or accessing any website i.e. driver.get() and driver.navigate().to().

These two methods are examples of Method Overriding.

o The get() method does not load the web page completely if user are going to do some other operation after loading a page. This is the reason for which get() is faster than navigate().

o Using get() method, you can not traverse back and forward whereas Navigate() supports

back and forth traversal of a web page using navigate().forward() and navigate().back().

Top Ten Test Automation Frameworks

Top Ten Test Automation Frameworks mostly used for automation projects:

1. Robot Framework:

Robot Framework is the best choice in case user want to use a python test automation framework for their test automation efforts. The Robot Framework is Python-based, but you can also use Jython(Java) or IronPython(.NET). The Robot Framework uses a keyword-driven approach to make tests easy to create.Robot Framework can also test MongoDB, FTP, Android, Appium, and more. It has many test libraries including the Selenium WebDriver library and other useful tools. It has a lot of API's to help make it as extensible as possible. The keyword approach used by Robot Framework is great for testers who are already familiar with other vendor-based, keyword-driven test tools, making the transition to open source much easier for them.

2. WebDriverIO:

WebdriverIO is an automation test framework based in Node.js. It has an integrated test runner and you can run automation tests for web applications as well as native mobile apps. Also, it can run both on the WebDriver protocol and Chrome Devtools protocol, making it efficient for both Selenium Webdriver based cross-browser testing or Chromium based automation. As WebDriverIO is open-source, you get a bunch of plugins for your automation needs. 'Wdio setup wizard' makes the setup simple and easy.

3. Citrus:

Citrus is an open-source framework with which you can automate integration tests for any messaging protocol or data format. For any kind of messaging transport such as REST, HTTP, SOAP, or JMS, Citrus framework will be suited for test messaging integration. If you need to interact with a user interface and then verify a back-end process, you can integrate Citrus with Selenium. For instance, if you have to click on a "send email" button and verify on the back end that the email was received, Citrus can receive this email or the JMS communication triggered by the UI, and verify the back-end results, all in one test.

4. Cypress:

Cypress is a developer-centric test automation framework that makes test-driven development (TDD) a reality for developers. Its design principle was to be able to package and bundle everything together to make the entire end-toend testing experience pleasant and simple. Cypress has a different architecture than Selenium; while Selenium WebDriver runs remotely outside the browser, Cypress runs inside of it. This approach helps in understanding everything that happens inside and outside the browser to deliver more consistent results. It does not require you to deal with object serialization or over-the-wire protocols while giving you native access to every object. Cypress can synchronously notify you of every single thing that happens inside the browser as you're pulling your app into it so that you have native access to every DOM element. It also makes it easy to simply drop a debugger into your application, which in turn makes it easier to use the developer tools.

5. Selenium:

One of the most popular open-source test automation frameworks for web apps. Selenium also serves as a base for a lot of other testing tools as it has cross-platform and cross-browser functionality. Selenium supports a wide range of programming languages such as Java, C#, PHP, Python, Ruby, etc. It is easy to maintain as it has one of the largest online support networks. Selenium is highly extendable through a wide range of libraries and APIs to meet everyone's needs and requirements. Selenium is preferred by testers as it is possible to write more advanced test scripts to meet various levels of complexity. It provides a playback tool for test authoring without the need to learn a specific scripting language.

6. Cucumber:

It is a cross-platform behavior-driven development (BDD) tool that is used to write acceptance tests for web applications. Cucumber is quick and easy to set up execution and allows reusing code in the tests. It supports languages like Python, PHP, Perl, .NET, Scala, Groovy, etc. Automation of functional validation in an easily readable and understandable format. One good feature is that both specification and test documentation are uploaded in a single up-to-date document. Cucumber makes it easy for the business stakeholders, who are not familiar with testing, as they can easily read the code as test reports are written in a business readable English. The code can be used together with other frameworks like Selenium, Watir, Capybara, etc.

7. Gauge:

It is an open-source tool agnostic test automation framework for Mac, Linux, and Windows. People who work on TDD and BDD will appreciate Gauge's focus on creating living/executable documentation. Specs – the Gauge automation tests are written using a markdown language with C#, Java, and Ruby within your existing IDEs like Visual Studio and Eclipse. Gauge's functionality can also be extended with its support of plugins. It was developed as a BYOT (Bring Your Tool) framework. So you can use Selenium or you can use anything else for driving your tests UI or API tests. If you want a readable non-BDD approach to automation, you should try Gauge.

8. Serenity:

If you are looking for a Java-based framework that integrates with behavior-driven development (BDD) tools such as Cucumber and JBehave, Serenity might be the tool for you. It's designed to make writing automated acceptance and regression tests easier. It also lets you keep your test scenarios at a high level while accommodating lower-level implementation details in your reports. Serenity acts as a wrapper on top of Selenium WebDriver and BDD tools. It abstracts away much of the boilerplate code you sometimes need to write to get started which makes writing BDD and Selenium tests easier. Serenity also offers plenty of built-in functionality, such as handling running tests in parallel, WebDriver management, taking screenshots, managing state between steps, facilitating Jira integration, all without having to write a single line of code.

9. Carina:

Carina is built using popular open-source solutions like Appium, TestNG, and Selenium, which reduces dependence on a specific technology stack. You can test mobile applications (native, web, hybrid), WEB applications, REST services, and databases. Carina framework supports different types of databases like MySQL, SQL Server, Oracle, PostgreSQL, providing the amazing experience of DAO layer implementation using MyBatis ORM framework. It supports all popular browsers and mobile devices and it reuses test automation code between IOS/Android up to 80%. API testing is based on the Freemarker template engine and itprovides great flexibility in generating REST requests. Carina is cross-platform and tests may be easily executed both on Unix or Windows OS.

10. EarlGray:

Developers often face difficulty with some of the existing test automation framework in synchronization of the app and the instrumentation. Also, executing tests on apps as synchronized and advanced only when UI elements are visible on the screen has caused issues for many developers. Google EarlGrey has built-in synchronization that makes test scripts wait for UI events to occur before the script tries to interact with the UI of the app. This type of implementation makes the test script concise as all steps of the test script shows how the test will proceed and UI gets synchronized with it. One more key aspect of EarlGrey is that all actions on UI elements happen only on visible elements. This provides a fast and robust approach to ensure UI testing goes through as clicks, gestures and other user interactions do not get done if the UI element is not fully shown.

Framework in Selenium

- Selenium framework is a code structure that is used to make code maintenance simpler and readability better. It helps in dividing the entire code into smaller parts of the code in a systematic way, which test a particular functionality.
- Without framework at starting, we had written complete code and data in the same place that was neither reusable nor readable. But using selenium framework, we can create the structure of code in better ways.
- We can keep "test data" separate from the actual test cases which help to analyze and update data easily. It increases code readability, helps to get the desired results in a systematic way, and reduces script maintenance cost.
- Without a framework, a single test case may have a million lines of code and it will test the entire functionality of the application. Now, suppose user need to modify any functionality later time so is it possible to read the entire code again for a particular functionality? Answer will be No because reading so huge test case will be tough and boring, and you also need lots of time to analyze the entire code.
- But when we use selenium framework structure, it breaks the entire code into small parts. These smaller parts can be easily readable if we need to modifying and can perform a particular function.

Modular Driven Framework

In this Framework, the tester can create test scripts module wise by breaking down the whole application into smaller modules as per the client requirements and create test scripts individually.

Pros: Modular driven framework ensures the division ofscripts that leads to easier maintenance and scalability.User can write independent test scripts.

Cons: The modular driven framework requires additional time in analyzing the test cases andidentifying reusable flows.

Data Driven Framework in Selenium

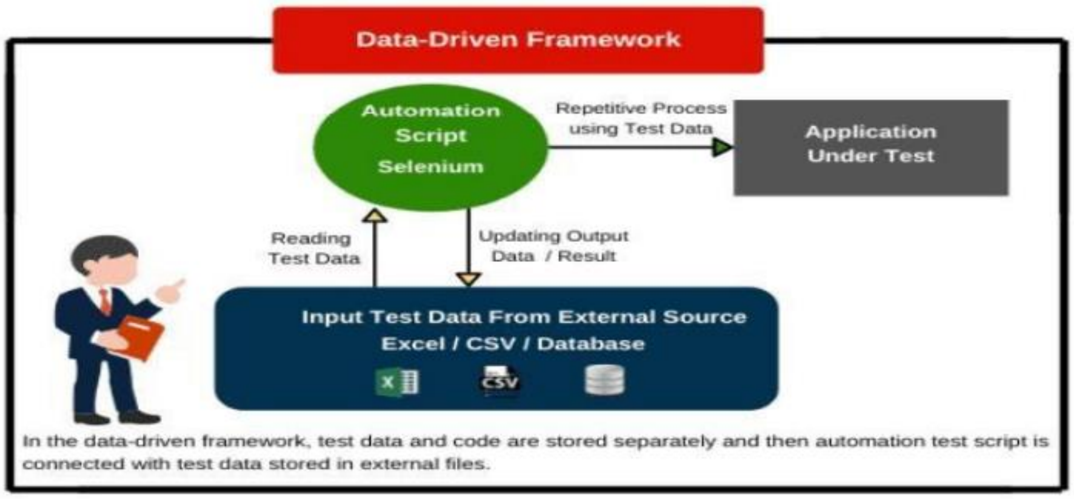
A Data Driven Framework in Selenium is a technique of separating the "data set" from the actual "test case" (code). Since the test case is separated from the data set, user can easilymodify the test case of a particular functionality without making changes to the code. Data Driven framework is used to drive test cases and suites from an external data feed. The data feed can be data sheets like xls, xlsx, and csv files.

• Say, user has to modify the code for login functionality, they can modify just the login functionality instead of having to modify any other feature dependent on the same code.

• This example will demonstrate how to read the data from excel files and perform data driven testing using Selenium. WebDriver does not directly support data reading of excel files. Therefore, one needs to use a plugin such as Apache POI for reading/writing on any Microsoft office document.

• To download Apache POI Jar files, navigate to https://poi.apache.org/download.html.

Download the zip file and place them along with the set of Selenium JARs and configure your build path. The co-ordination between the main code and data set will be taken care by TestNG Data Providers, which is a library that comes as a part of the Apache POI JAR files.


Data-Driven Framework
Automation Script Selenium
Repetitive Process using Test Data
Application Under Test
Reading Test Data
Updating Output Data / Result
Input Test Data From External Source
Excel / CSV / Database
CSV
In the data-driven framework, test data and code are stored separately and then automation test script is connected with test data stored in external files.

Why Data Driven Tests important?

They reduce the cost of adding new tests and changing them when your business rules change. This is done by creating parameters for different scenarios, using data sets that the same code can run against.

They help to identify what data is most important for tested behavior. By separating first-class scenario data into parameters, it becomes clear what matters most to the test. This makes it easy to remember how something works when developers need to change it.

What are the major Advantages of Data Driven Testing Framework?

- Allows testing of the application multiple sets of data values during regression testing
- Separates the test case data from with the executable test script
- Allows reusing of Actions and Functions in different tests
- Generates test data automatically. This is helpful when large volumes of random test data are necessary
- Results in the creation of extensive code that is flexible and easy to maintain
- Let's developers and testers separate the logic of their test cases/scripts from the test data
- Allows execution of test cases several times which helps to reduce test cases and scripts
- It does not let changes in test scripts affect the test data.
- By incorporating data-driven testing using Selenium, testers can refine their test cases for more efficient execution. This shortens timelines, makes their lives easier and results in more thoroughly tested and better-quality software.

Major disadvantages of Data Driven Framework

- This calls for great expertise of scripting language required by the automation tool.
- For every test case user need many data-files. According to the number of screens being accessed user may have many data-inputs and may require many verifications.
  Hence the test case needs to keep the data-files in different directories.
- In case for creating & maintaining the data-files, user use text editor like Notepad, extra care is needed in having the desired format needed by the functions or scripts which would process the concerned files; otherwise, they shall get script-processing errors due to incorrect data-file format or its content

Class: 22

Agenda

- **Reading and Writing Data in Excel Sheet through Data Driven Framework**
- **Keyword Driven Framework in Selenium**
- **Data Driven Framework in Selenium**

Apache POI is an open-source library developed and distributed by Apache Software Foundation to design or modify Microsoft Office files using Java program. It is a popular API that allows working around excel files using Java Programs. In short, user canread and write MS Excel files using Java. Apache POI is your Java Excel solution.

User need to us use XSSF if they need to read or write an OOXML Excel file using Java (XLSX).

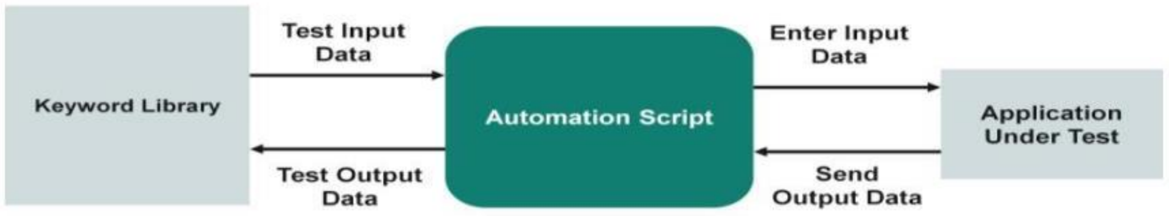
It has many predefined methods, classes, and interfaces.

Prerequisites to implement Data Driven Framework:

- Eclipse IDE
- TestNG
- Selenium jars

Keyword Driven Framework in Selenium

Keyword Driven framework is a technique in which all the operations & instructions to be performed are written separately from the actual test case. The similarity it has with Data Driven framework is that the operations to be performed is again stored in an external file like Excel sheet. Users can easily control and specify the functionalities they want to test.



Advantages of Keyword Driven Testing Framework

- A single keyword can be used across multiple test scripts, so the code is reusable.
- User can write the detailed test plan having desired inputs and verification data in the form of simple & convenient spreadsheets.
- In case some expert having expertise of scripting language used by the automation tool, can create utility scripts well before the creation of the detailed test plan, then the testing engineer is able to use the Automation Tool immediately through the spreadsheet input method & for this they need not master the concerned scripting language used by the automation tool.

- User becomes productive with the new testing tool far quickly, since they required to understand the format for use in the test plan & various desired Key Words. This permits deferment / rescheduling of more exhaustive training on new tool at some more convenient time in future.

Disadvantages of Keyword Driven Framework

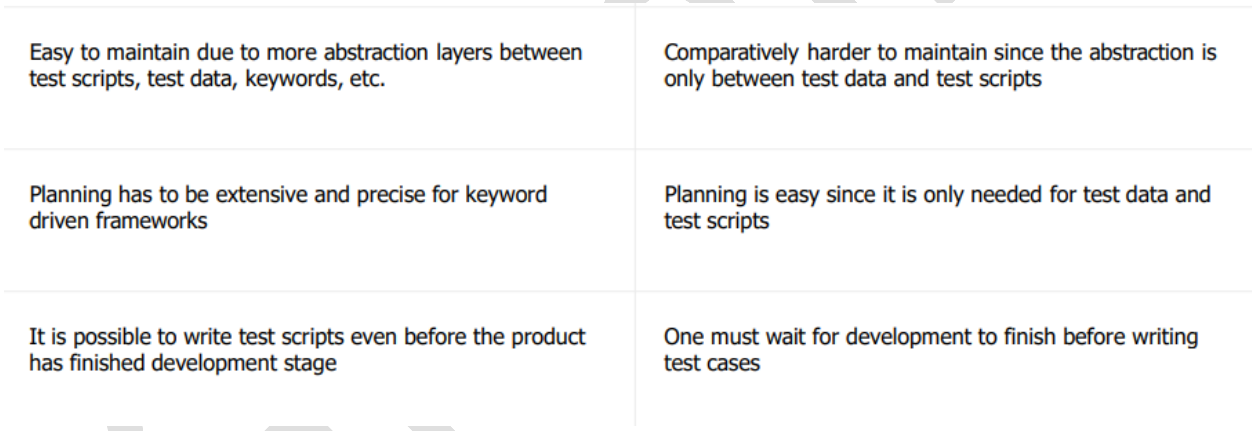
The initial cost of setting up the framework is high, and it is time-consuming & complex.

The tester is required to have great expertise over the scripting language of the test tool, so that they can successfully create customized functions and utilities required for the application. Although this statement is valid for all other methods as well.

The tester is required to learn several special formats & keywords for creating the customized utilities required for the application. This process can take considerable time & can have adverse effect on the test plan development.

However, these problems arise in the beginning only, and when the tester get enough acquaintance with the process, they are able to create test cases much faster.

Difference between Data and Keyword Driven Testing Framework



Easy to maintain due to more abstraction layers between test scripts, test data, keywords, etc.	Comparatively harder to maintain since the abstraction is only between test data and test scripts
Planning has to be extensive and precise for keyword driven frameworks	Planning is easy since it is only needed for test data and test scripts
It is possible to write test scripts even before the product has finished development stage	One must wait for development to finish before writing test cases

Class:23

- **Hybrid Framework in Selenium**
- **Introduction to Page Object Model**
- **Implementation and Set Up of POM with the help of a Maven Project**

Hybrid Framework in Selenium

- Hybrid Framework in Selenium is a concept where tester are using the advantage of both Keyword driven framework as well as Data driven framework. It is an easy-to-use framework which allows manual testers to create test cases by just looking at the keywords, test data and object repository without coding in the 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.

Advantages of hybrid automation framework

The Hybrid framework is built with several reusable modules / function libraries that are developed with the following features in mind:

Maintainability – Hybrid framework significantly reduces maintenance

effort

Re-usability – It allows to reuse test cases and library functions

Manageability - effective test design, execution, and traceability

Accessibility – easy to design, develop, modify and debug test cases while executing

Availability – Allows to schedule automation execution

Reliability – due to advanced error handling and scenario recovery

Flexibility – framework independent of system or environment under test

Measurability – customizable reporting of test results ensure the quality output

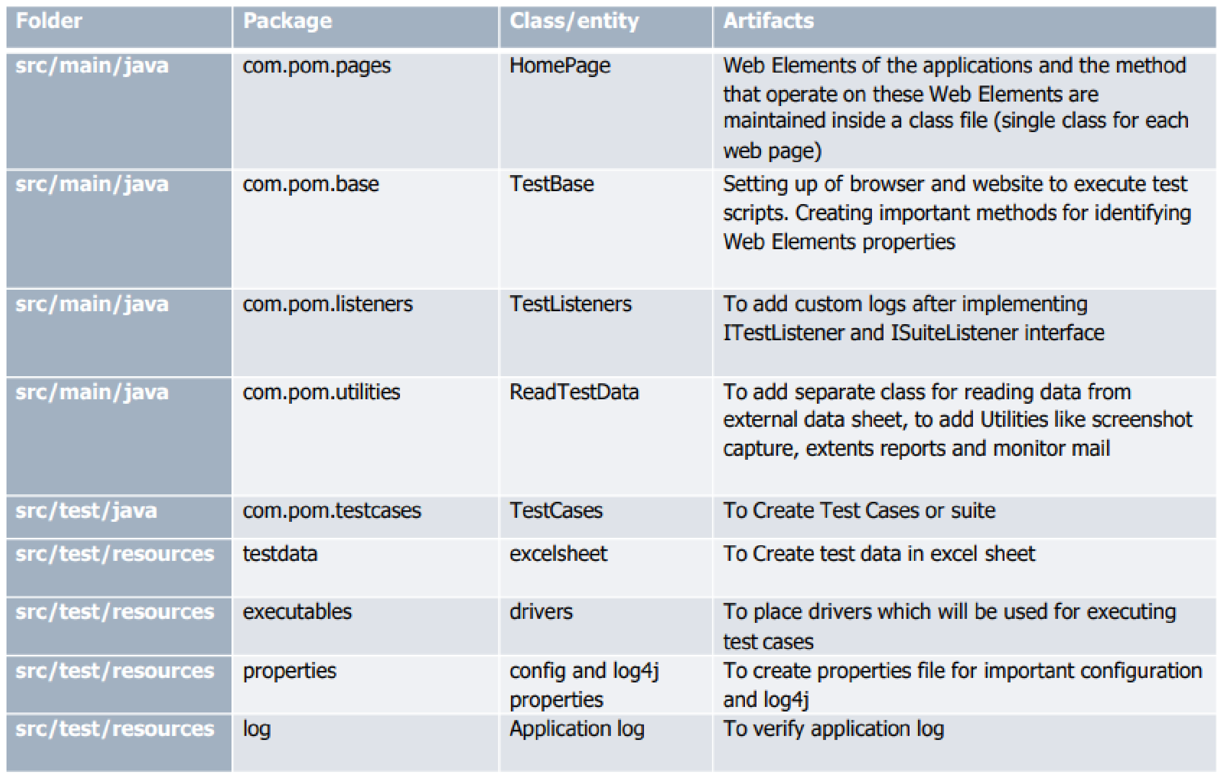
Introduction to Page Object Model

Page Object Model (POM) in Selenium is a design pattern that creates a repository of objects, such as buttons, input fields, images, links and other elements. The main objective of using POM in Selenium is to reduce code duplication and improve the maintenance of tests in the future. User will create a class file for each web page in Page Object Model framework. 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.

Advantages:

- Helps with easy maintenance: POM is useful when there is a change in a UI element or there is a change in an action. An example would be if a drop-down menu is changed to a radio button. In this case, POM helps to identify the page or screen to be modified. As every screen will have different java files, this identification is necessary to make the required changes in the right files. This makes test cases easy to maintain and reduces errors.
- Helps with reusing code: By using POM, user can use the test code for any one screen and reuse it in another test case. There is no need to rewrite code, thus saving time and effort.
- Readability and Reliability of scripts: When all screens have independent java files, one can easily identify actions that will be performed on a particular screen by navigating through the java file. If a change must be made to a certain section of code, it can be efficiently done without affecting other files.

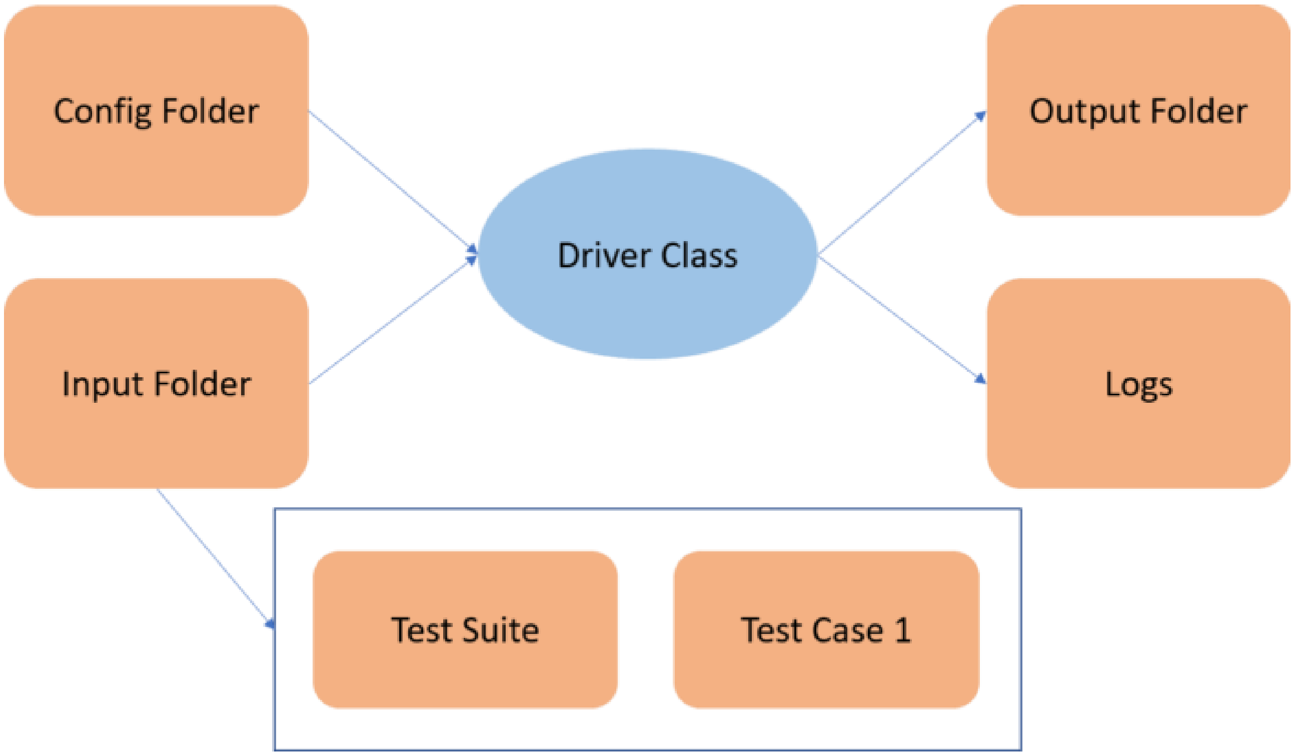
Page Object Model Architecture



Folder	Package	Class/entity	Artifacts
src/main/java	com.pom.pages	HomePage	Web Elements of the applications and the method that operate on these Web Elements are maintained inside a class file (single class for each web page)
src/main/java	com.pom.base	TestBase	Setting up of browser and website to execute test scripts. Creating important methods for identifying Web Elements properties
src/main/java	com.pom.listeners	TestListeners	To add custom logs after implementing ITestListener and ISuiteListener interface
src/main/java	com.pom.utilities	ReadTestData	To add separate class for reading data from external data sheet, to add Utilities like screenshot capture, extents reports and monitor mail
src/test/java	com.pom.testcases	TestCases	To Create Test Cases or suite
src/test/resources	testdata	excelsheet	To Create test data in excel sheet
src/test/resources	executables	drivers	To place drivers which will be used for executing test cases
src/test/resources	properties	config and log4j properties	To create properties file for important configuration and log4j
src/test/resources	log	Application log	To verify application log

Execution Flow of Hybrid Framework

The diagram below depicts the architectural design of the Hybrid framework:



- **Execution starts with reading the configuration file**. This file will reveal the location and name of the input file, as well as the location of the output file. This is the test suite

- **The input file is an Excel sheet.**

- It contains the first sheet as the Test Suite. The Test Suite sheet has four columns:
    a. Name of the test case
    b. Run Flag (whether to execute respective test cases or not)
    c. Comments
    d. Run Status

- **Name of the test case in the Test Suite sheet is the same as the name of the Test Case sheet**

- Each test case has 6 columns:
    a. Keyword
    b. Locator
    c. Argument
    d. Comment

e. Run Status
f. Return value

- Each row in a test case sheet represents a test step

- Comments are updated in the comment column if any test step fails or is skipped with the proper reason

- If a keyword returns some value, then that is updated in the Return value column

- Each step is executed in the sequential column. If one or more steps fail in a test case, the corresponding test case is also marked FAIL in the test suite

- Once all the test cases of a test suite are executed, and the corresponding status is updated in both sheets, a result file with a new name (timestamp appended) is saved in the Result folder

- Logs and screenshots are saved in the Output folder

Executing Hybrid Test Framework in Selenium

If you compare test case execution, the Hybrid framework is almost the same as its Keyword-Driven counterpart with only a couple of changes required.

Components of Hybrid Framework are as follows:

- Function Library
- Excel Sheet to store Keywords
- Design Test Case Template
- Object Repository for Elements/Locators
- Test Scripts or Driver Script

Class :24

Refer BDD CUCUMBER CHEAT SHEET

Class :25

Agenda

- **Introduction to TDD and BDD**
- **Cucumber Introduction**
- **Advantages of Cucumber Over Other Tools**
- **Prerequisite required for using Cucumber with Selenium**
- **Developing one simple Cucumber Framework from Scratch**
- **Add Code inside Step Definitions Class**

TDD Overview

TDD (Test-Driven Development) is an iterative development process. Each iteration starts with a set of tests written for a new piece of functionality. These tests are supposed to fail during the start of iteration as there will be no application code corresponding to the tests. In the next phase of the iteration, Application code is written with an intention to pass all the tests written earlier in the iteration. Once the application code is ready tests are run.

This helps us in many ways:

- User write the application code based on the tests. This gives a test first environment for development and the generated application code turns out to be bug-free.
- With each iteration, user write tests and as a result, with each iteration, userget an automated regression pack. This turns out to be very helpful becausewith every iteration user can be sure that earlier features are working.
- These tests serve as documentation of application behavior and reference for future iterations.

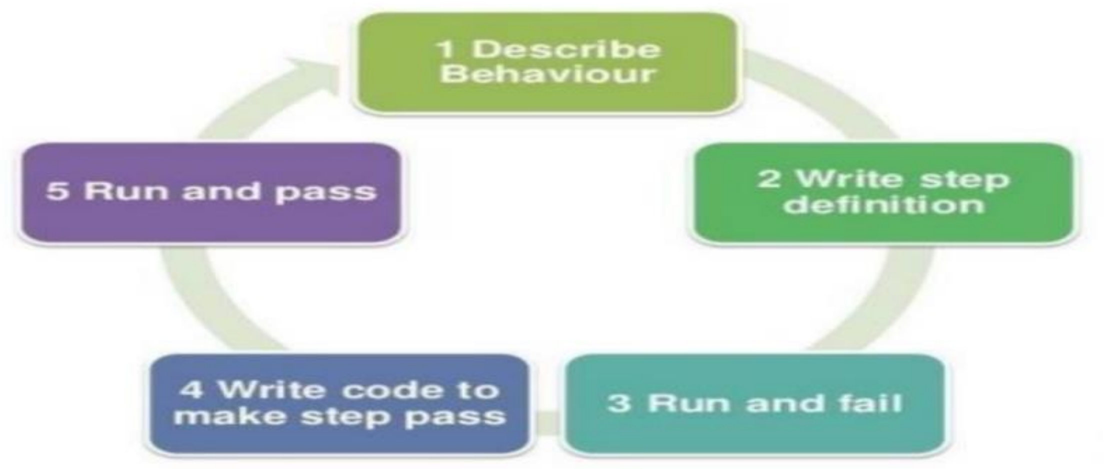
Behavior Driven Development

Behavior Driven testing is an extension of TDD. Like in TDD in BDD also user write tests first and the add application code. BDD gives user an opportunity to create test scripts from boththe developer's and the customer's perspective as well. So, in the beginning, developers,project managers, QAs, user acceptance testers and the product owner (stockholder), all get together and brainstorm about which test scenarios should be passed in order to call this software/application successful. This way they come up with a set of test scenarios. All these test scripts are in simple English language, so it serves the purpose of documentation also.

The major difference between TDD and BDD are:

- Tests are written in plain descriptive English type grammar
- Tests are explained as behavior of application and are more user-focused
- Using examples to clarify requirements
- This difference brings in the need to have a language that can define, in an understandable format

Features of BDD



Shifting from thinking in "tests" to thinking in "behavior"

- Collaboration between Business stakeholders, Business Analysts, QA Team and developers
- Driven by Business Value
- Extends Test-Driven Development (TDD) by utilizing natural language that non-technical stakeholders can understand
- BDD frameworks such as Cucumber or JBehave are an enabler, acting a "bridge" between Business & Technical Language
- BDD is popular and can be utilized for Unit level test cases and for UI level test cases.
- 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

Introduction to Cucumber

- A cucumber is a tool based on Behavior Driven Development (BDD) framework which is used to write acceptance tests for the web application. It allows automation of functional validation in easily readable and understandable format (like plain English) to Business Analysts, Developers, Testers, etc.
- Cucumber feature files can serve as a good document for all. There are many other tools like JBehave which also support BDD framework. Initially, Cucumber was implemented in Ruby and then extended to Java framework. Both the tools support native JUnit.
- Cucumber reads the code written in plain English text (Gherkin Language) in the feature file
- It finds the exact match of each step in the step definition (a code file).
- The piece of code to be executed can be different software frameworks like Selenium, Ruby on Rails, etc. Not every BDD framework tool supports every tool.
- This has become the reason for Cucumber's popularity over other frameworks, like JBehave, JDave, Easyb, etc.

Cucumber supports over a dozen different software platforms like:

- Ruby on Rails
- Selenium
- PicoContainer
- Spring Framework
- Watir

Advantages 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.

Prerequisite required for using Cucumber

User need the following items before starting cucumber:

Step 1: Download and install the Java platform on user machine

Step 2: Download and install Eclipse IDE

Step 3: Download Cucumber Eclipse Plugin:

a. In the eclipse, navigate to Help > Install New Software. Copy the URL "http://cucumber.github.io/cucumbereclipse/update-site/" and press Enter.

b. User would see a checkbox named "Cucumber Eclipse Plugin", Select the checkbox 'Cucumber Eclipse Plugin'.

c. Click 'Next'

d. Again click 'Next' and Accept the license terms.

e. Click Finish

f. Click 'Install anyway'

g. Click 'Restart Now'

Step 4: Create a Maven Project in Eclipse

Step 5: Open the pom.xml file in eclipse and the below dependency after navigating to Maven Repository

"https://mvnrepository.com/"

a. cucumber-java

b. cucumber-core

c. cucumber-junit

d. cucumber-picocontainer

e. cucumber-gherkin

Note: io.cucumber is the new version and info.cukes are the older version. Wherever user can see both io.cucumber and info.cukes, they need to go with the version io.cucumber as its artifact id

Cucumber framework mainly consists of three major parts – Feature File, Step Definitions, and the Test Runner File.

1. Feature File

A standalone unit or a single functionality (such as a login) for a project can be called a Feature. Each of these features will have scenarios that must be tested using Selenium integrated with Cucumber. A file that stores data about features, their descriptions, and the scenarios to be tested, is called a

Feature File.

Cucumber tests are written in these Feature Files that are stored with the extension – ".feature".

A Feature File can be given a description to make the documentation more legible.

Example:

The Login function on a website

Feature File Name: userLogin.feature

Description: The user shall be able to login upon entering the correct username and password in the correct fields. The user should be directed to the homepage if the username and password entered are correct.

Keywords such as GIVEN, WHEN, and THEN used to write the test in Cucumber are called Annotations.

GIVEN user navigates to login page by opening Firefox

WHEN user enters correct <username> AND <password> values

THEN user is directed to the homepage

2. Step Definitions

Now that the features are written in the feature files, the code for the related scenario has to be run. To know which batch of code needs to be run for a given scenario, Steps Definitions come into the picture. A Steps Definitions file stores the mapping data between each step of a scenario defined in the feature file and the code to be executed.

Step Definitions can use both Java and Selenium commands for the Java functions written to map a feature file to the code.

Example:

import cucumber.api.java.en.Given;

import cucumber.api.java.en.Then;

```
import cucumber.api.java.en.When;

public class Steps

{

@Given("^user navigates to the login page by opening Firefox$")

//Code to Open Firefox Browser and launch the login page of application to define the GIVEN step of
the feature

@When("^user enters correct username and password values$")

//take inputs for username and password fields using find element by xpath. Put the correct
username and password values as inputs to define the WHEN step of the feature

@Then (“^user gets directed to homepage$”)

//Direct to the Homepage of the application as a result of correct username and password inputs in
the WHEN step. This would define the THEN step of the feature
```

Class:26

- **Create Multiple Scenarios**
- **Implements undefined Steps**
- **Create Multiple Feature Files**
- **Create Multiple Scenarios**
- **Scenario Outline**
- **Adding Examples**

Scenario Outline

Scenario includes all the possible circumstances of the feature and test scripts for these circumstances. The keyword "Scenario" represents a scenario in Gherkin language. One feature can have multiple scenarios, and each scenario consists of one or more steps.

Cucumber Scenario Outline in Gherkin Based on Gherkin Reference, the Scenario Outline keyword can be used to repeat the same steps with different values or arguments being passed to the step definitions. This is helpful if user want to test multiple arguments in the same scenario.

When to use scenario outlines user want their scenario outlines to describe behaviors or examples that are important to the business, not detail every boundary or test case. Adding too many rows simply creates more work and obscures where the value is to the business.

All scenario outlines are followed by Examples part that contains the set of data that has to be passed during the execution of tests. Pipe symbol (|) is used to specify one or more parameter values in a feature file.

In a single execution, Scenario is executed only once whereas Scenario outline (For similar data trace) can be executed multiple times depending upon the data provided as Example

Hands on Project practice Eclipse

Class :27 & 28

Agenda

- **Cucumber Tags**
- **Execute Scenarios with Tags**
- **Exclude Scenarios with Tags**
- **OR & AND Tags in Cucumber**
- **Cucumber Hooks**
- **Tagged Hooks In Cucumber**
- **Setting Order or Priority of Cucumber Hooks**
- **Cucumber Report**
- **HTML Report**
- **JSON Report**
- **XML Report**
- **Extent Report**
- **Steps to Generate Extent Report using Cucumber**
- **Maintain Backup of All Generated Extent Reports**

Tags in Cucumber

- It might look very simple when user just have one, two, or maybe five scenarios in a feature file. However, in real life it does not happen. For each feature under test, user may have 10, 20, or may be a greater number of scenarios in a single feature file. They may represent different purpose (Smoke/Integration/End to End/Regression test), different prospective (Developer/QA/BA), different status (Ready for execution/Work in progress/defect fixes), etc.
- Cucumber has already provided a way to organize user scenario execution by using tags in feature file. User can define each scenario with a useful tag. Later, in the runner file, they can decide which specific tag (and so as the scenario(s)) user want Cucumber to execute. Tag starts with "@". After "@" user can have any relevant text to define their tag.
- Suppose, there are two or more scenarios in a feature file. User want to execute only one scenario as part of smoke test. So, first thing is to identify that scenario and second is to tag it with "@SmokeTest" text at the beginning of the scenario.
- There is no limit in defining tags within the feature file. Based on user need, they can derive tags to be used and scenarios to be executed.
- Tag can also be defined at a feature level. Once user define a tag at the feature level, it ensures that all the scenarios within that feature file inherits that tag. Depending on the nature of the scenario, user can use more than one tag for the single feature. Whenever Cucumber finds an appropriate call, a specific scenario will be executed.
- For excluding tag, user can use "not " in JUnit runner class to exclude smoke test scenario. It will look like the following.

```
@RunWith(Cucumber.class)

@Cucumber.Options(plugin = ("pretty"), tags = (“not @SmokeTest"))
```

Tags using AND/OR in Cucumber

- Sometimes, requirements are complicated, it will not always simple like executing a single tag. It can be complicated like executing scenarios that are tagged either as @SmokeTest or @RegressionTest. It can also be like executing scenarios that are tagged both as @SmokeTest and @RegressionTest. Cucumber tagging gives us the capability to choose what we want with the help of ANDing and ORing.
- Tags that separated with “or” are ORed. OR means scenarios that are tagged either as @SmokeTest OR @RegressionTest.
- Tags which are passed with “and” separator are ANDed.When both the tags will match only, scenarios will be executed.

Hooks in Cucumber

- Multiple prerequisites are required at the start of test execution for most of the times. Right from setting up the web driver, browser settings to cookies, navigating to the specific URL, etc.
- Similarly, some steps must be performed after executing the test scenarios, like quitting driver, clearing browser cookies, generating reports, etc. Such cases can be easily handled using one particular type of Cucumber annotations, namely Cucumber Hooks.
- Cucumber hooks are blocks of code that runs before or after each scenario. It can be defined anywhere in project or step definition layers using methods @Before, @After. Cucumber hooks Annotations allow us to manage better code workflow and help in reducing code redundancy. Cucumber hooks are used in a situations where prerequisite steps before testing any test scenario is performed.
- User might have worked across different TestNG annotations, like BeforeTest, BeforeMethod, BeforeSuite, AfterTest, AfterMethod, AfterSuite, etc. Unlike TestNG, Cucumber provides only two hooks, i.e., Before and After.

@Before Hook: It will execute before every scenario.

Example

```
@Before

public void setUp() {

System.out.println("Starting the test");

}
```

@After Hook: It will execute after every scenario.

```
@After

public void tearDown () {

System.out.println("Closing the test");

}
```

Tagged Hooks in Cucumber

- Tagged Hooks are basically the problem solvers when user need to perform different Before and After actions for different scenarios. To explain in a simpler way, think user have 5 different tags like Sanity, Regression, Integration etc., which need to be tested with different URLs. This is where Tagged hooks help you achieve that.
- Now we will create the hooks file and define the tagged hooks in it. This can be done using the

@Before or the @After hook followed by the tag name of the scenario as shown in the code below:

```
@Before("@Smoke")

public void beforeSmoke(){

System.out.println("This will be executed before any smoke test cases");

}

@After("@Smoke")

public void afterSmoke(){

System.out.println("This will be executed after any smoke test cases");

}

@Before("@Integration")

public void beforeIntegration(){

System.out.println("This will be executed before any Integration test cases");
}
@After("@Integration")
public void afterIntegration(){
System.out.println("This will be executed after any Integration test cases");
}
```

Setting Order or Priority of Cucumber Hooks

- In case user have already worked on TestNG, they must be familiar with the priority of tests and execution order. Similarly, Cucumber hooks can also be executed as per order.
- Let's, consider an example with a hooks file consisting of two @After hooks and two @Before hooks.
- User will set the order of the hooks as per our requirement by simply specifying the order as in the hooks file code below:

```
@Before(order=0)
public void setUp() {
System.out.println("This will execute first--Before");
}
@After(order=1)
public void tearDown() {
System.out.println("This will execute first--After");
}
@Before(order=1)
public void setUpTwo() {
System.out.println("This will execute second--Before");
}
@After(order=0)
public void afterOne() {
System.out.println("This will execute second--After");
}
```

Reports in Cucumber

- During test execution (Either Manual or Automation), it is always required to understand the out put of the execution. The execution output should display in specific format, which immediately depicts the overall results of the execution. Hence, our framework also should have the same capability to create output or generate test execution reports.
- It is essential to know, how better user can generate our Cucumber test reports. As Cucumber is a BDD framework, it does not have a fancy reporting mechanism. In order to achieve this, Cucumber itself has provided a nice feature to generate reports.
- Cucumber.io has developed a free cloud-based service for sharing reports. Cucumber reporting service allows to configure cucumber to publish results to the cloud which can be accessed from browser throughout the organization.
- Cucumber uses reporter plugins to produce reports that contain information about which scenarios have passed or failed.

Pretty Report: The first plugin, is Pretty. This provides more verbose output. To implement this, just specify plugin = "pretty" in CucumberOptions.

@CucumberOptions(plugin = ("pretty"))

Monochrome Mode Reporting: If the monochrome option is set to false, then the console output is not as readable as it should be. In case the monochrome is not defined in Cucumber Options, it takes it as false by default.

@CucumberOptions(monochrome = true);

HTML, JSON, XML and JUNIT Reports in Cucumber

HTML Report : To generate custom report in html format, include below plugin in Runner class,

plugin = {

"pretty", “html:target/html-reports/report.html” }

After execution Cucumber will generate HTML report under target/html-reports/ folder.

JSON Report: To generate custom report in json format include below plugin in Runner class,

plugin = {

"pretty", "json:target/json-reports/report.json“ }

XML Report: To generate custom report in xml format include below plugin in Runner

class,

plugin = {

"pretty", "junit:target/xml-reports/report.xml“ }

JUNIT Report :

In case of using junit, reports can be generated in a separate file depends

on runner and by providing below plugin details in runner class,

plugin = { “pretty”,

“html:target/html-reports/report.html”,

“junit:target/junit-reports/”

}

Extent Report

In the previous video you already learnt how to generate reports in HTML, JSON, XML and JUNIT Format. However, these reports might not be appropriate sometime to share with respective stakes holder as demand is complete exhaustive reports with proper graphs, pie-chart and more granular details.

Extent Report is a powerful, open-source library used in testing automation frameworks for generating beautiful and user-friendly HTML reports. It allows the user to customize the report template by using custom CSS, JSON, or XML. It can integrate with almost all the major testing frameworks like JUnit, TestNG, etc. Extent reports are HTML-based documents that can carry detailed information about the test executed along with custom logs, screenshots and use a pie chart to represent an overview of the test.

Extent reports cannot directly integrate with the Cucumber framework. However, in case user want to generate extent reports for their Cucumber features, user will need to use some adapter plugin. Additionally, this plugin will allow the Extent report to recognize and capture scenarios and features present in the framework. It is where the grasshopper cucumber adapter plugin comes into the picture. The plugin is built on top of the Extent Report and allows user to quickly generate extent reports for their cucumber framework. Moreover, this plugin helps in reducing the pain of implementing the reporting in user framework.

Steps to Generate Extent Report using Cucumber

- Add grasshopper extent report adapter plugin to the Maven project:Navigate to the "https://mvnrepository.com/", type "extentreports" in the search box and select latest "ExtentReports Cucumber7 Adapter". Add the maven dependency in the Maven project pom.xml file

- Add Extent Report Library to the project:Again, navigate to the "https://mvnrepository.com/", type "extent-pdf-report" in the search box and select latest "Extent PDF Report". Add the maven dependency in the Maven project pom.xml file.

- Create a file named "extent.properties" in same folder where feature files were located. Addthe following lines to the file.

  ```
  extent.reporter.spark.start=true

  extent.reporter.spark.out=target/automation-report.html
  ```

- Add the following lines in user TestRunner.java class

```
plugin = { "pretty",

"com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:" }
```

After completing all the setup and configuration run TestRunner class and then refresh the

project. A beautiful HTML report will be generated in the target folder.

How to Maintain Backup of All Reports

- As of now we have learned how user can generate an extent report in Cucumber Junit. in case of multiple times execution, it will keep on overriding the previous report once the new report creates. Usually, user need to maintain the backup of all the reports generated from previous tests. Hence, user need to save each report with a different report name or folder name.
- With the Extent reporter plugin adapter, it's pretty easy to create reports in the different folder names. User need to add two settings to their extent.properties file; basefolder.name and basefolder.datetimepattern. The values assigned to these will merge to create a folder name. Consequently, a report will generate inside that.
- The value for the basefolder.datetimepattern should be in a valid datetime format.

  ```
  basefolder.name=target/ExtentReport

  basefolder.datetimepattern=dd-MMM-YY HH-mm-ss
  ```

Class: 29 & 23

What materials providing as bonus to this course

Resume Preparation

Tips to Get More Calls For Resume

Tips To survive in IT company

What you need to learn once you are in to Software company