**Generic Utilities:**

**1. What is Generic Utility? How is it related with framework?**

Ans. Generic Library is a component of our framework which contains classes and interfaces composed of generic methods which can be used in the test scripts

In our framework we create a package named as “GenericUtilities” in src/main/java folder

Classes and interfaces are created inside this package which contains generic methods designed by framework developers

Certain actions such as maximize window, setting waits, scroll page, handling dropdowns, reading from properties file, reading from excel files etc, are common to all the test scripts. So instead of repeating the entire code for such methods in every test script, we can create and store them in generic library/utility as methods and utilize them as and when needed

This makes maintenance, modifications and debugging of scripts easier.

For this purpose, we create generic methods and parameterize them to accept value from calling method and to return value to the caller, we can use return statements inside these methods

**2. Write any 2 methods From Excel Utility**

Ans.

//Method 1: To read data from Excel File

**public** String readDataFromExcelSheet(String sheetName, **int** rowNum, **int** cellNum)

**throws** EncryptedDocumentException, IOException {

FileInputStream fis = **new** FileInputStream(“.\\src\\test\\resources\\TestData.xlsx.xlsx”);

Workbook wb = WorkbookFactory.*create*(fis);

Cell ce = wb.getSheet(sheetName).getRow(rowNum).getCell(cellNum);

String value = ce.getStringCellValue();

**return** value;

}

//Method 2: To write Data to Excel file

**public** **void** writeDataIntoExcelSheet(String sheetName, **int** rowNum, **int** cellNum, String cellValue)

**throws** EncryptedDocumentException, IOException {

FileInputStream fis = **new** FileInputStream(IConstantsUtility.***ExcelFilePath***);

Workbook wb = WorkbookFactory.*create*(fis);

Cell ce = wb.getSheet(sheetName).getRow(rowNum).createCell(cellNum);

ce.setCellValue(cellValue);

FileOutputStream fos = **new** FileOutputStream(IConstantsUtility.***ExcelFilePath***);

wb.write(fos);

wb.close();

}

**3. write An 2 re-usable Methods to Handle drop down**

Ans.

//Method 1: This method will select option from dropdown with matching value

public void handleDropDown(WebElement element, String value){

Select sel = new Select(element);

sel.selectByValue(value);

}

//Method 2: This method will select value by visible text

public void handleDropDown(WebElement element,String text){

Select sel = new Select(element);

sel.selectByVisibleText(text);

}

**4. write any 3 generic method to Handle Synchronization in your framework**

Ans.

//Method 1: This method is used to give implicit wait of 20 seconds for every web element to load

public void waitForPageLoad(WebDriver driver){

driver.manage().timeOuts().implicitlyWait(Duration.ofSeconds(20));

}

//Method 2: This method will wait for a particular element to become clickable

public void waitForElementToBeClickable(WebDriver driver, WebElement ele){

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(20));

wait.until(ExpectedConditions.elementToBeClickable(ele));

}

//Method 3: This method will wait for a particular element to become visible

public void waitForElementToBeClickable(WebDriver driver, WebElement ele){

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(20));

wait.until(ExpectedConditions.visibilityOf(ele));

}

**5. write any 3 generic methods from Web driver Utility**

Ans.

//Method 1: This method will maximize the browser

public void maximiseWindow(WebDriver driver){

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

//Method 2: This method will perform mouse hover action on an element

public void mouseHoverAction(WebDriver driver, WebElement ele){

Actions act = new Actions(driver);

Act.mouseHover(ele).perform();

}

//Method 3: This method will perform right click on a particular element

public void rightClickAction(WebDriver driver, WebElement ele){

Actions act = new Actions(driver);

Act.contextClick(ele).perform();

}

**6. Explain the advantages of using generic utilty component**

Ans. Advantages of Generic Utility:

1. Code maintenance becomes easier
2. Code modification becomes easy as the methods are kept in designated files. And any changes made to methods in Generic Utility classes are reflected automatically in all test scripts.
3. Test script development becomes faster since we only need to call the pre-defined methods
4. Repetition of code in test script is avoided
5. Debugging becomes easier as length of code in test scripts is less

**7. What are the classes and interfaces in generic utility.**

Ans. Generic Utility in our framework consists of following classes/interfaces:

1. *WebDriverUtility class* – A class consisting of generic methods related to WebDriver, such as, maximiseWindow() , minimizeWindow(), switchToFrame(), handleDropDown(), mouseHover(), takeScreenshot() etc.
2. *PropertyFileUtility class* – This class consists of methods related to Property file handling such as readDataFromPropertyFile() which can be used to read Common data.
3. *ExcelFileUtility class* – This class consists of methods associated with reading and writing test data to/from Excel file.
4. *JavaUtility class* – This class consists of generic methods related to Java library such as generateRandomNumber(), getSystemDataAndTime() etc
5. DatabaseUtility class – Consists of methods related to reading/writing to database
6. BaseClass – This class consist of all basic configuration annotations. This class is extended by every test script as it contains all basic configurations for our tests such as @BeforeClass, @BeforeMethod, @AfterMethod etc
7. *ListenerImplementationClass* – This class provides implementations to all abstract methods of ITestListener interface.
8. *IConstantsUtility interface* – Consists of common file paths used across classes. Such as PropertyFilePath, ExcelFilePath etc
9. RetryAnalyzerImplementation Class – Provides implementation to retry() method of IRetryAnalyzer interface.

**Object Repository**

**1. What is Object Repository? Why Object Repo?**

Ans.

* Object Repository is a component in our framework provided by Page Object Model (POM) concept design pattern
* Object repository is a storage unit where we store all the web elements along-with their locators in POM classes
* These object repositories are created Page Wise, i.e., for each web page we create a separate POM class and this class is used to store the web elements and their locators available in the web page
* Object repository is used because as per the automation rule, we should never hardcode the web elements and their locators in test scripts. This is important because if we repeat the locators in every test script, then to accommodate the UI changes with every new build, we will have to update all the locators in multiple test scripts
* To avoid this, the objects/elements are stored separately in object repository

**2. Why POM classes are prefered for Object repository?**

Ans.

* Pom classes are preferred for Object Repository because as per the POM design pattern, web elements are maintained page wise. Since for all web pages, all the elements and their locators will be stored in separate class files, the maintenance and modification of these elements becomes easier.
* POM classes are also preferred as they avoid StaleElementReferenceException. This exception occurs in situation where we identify a web element using a driver reference. But when we try to perform any action of that identified element, the element is either no longer attached to the DOM or the page got refreshed or the element reference may be old. In this situation, we get StaleElementReferenceException
* POM avoids this exception. In POM we use @FindBy, @FindAll and @FindBys annotations to identify the elements. And these elements are initialized with updated latest driver reference when the object is created.

**3. What is POM and write all the rules of POM?**

Ans.

* POM stands for Page Object Model
* POM is a java design pattern preferred by google to develop object repository. As per POM design pattern, we create separate classes for all the web pages of an application. The objects/elements of each page are declared, initialized and utilised in their respective POM classes.
* Rules of POM are as follows:
* For every web page, there should be a designated POM class
* Name of the class should be same as the title/name of the web page and should end with the word “Page”
* In every POM class, first we do element declaration. In this, we identify each web element using @FindBy, @FindBys or @FindAll annotations such as @FindBy(locatorStrategy=”value”) and declare the webElement as private
* Then we perform element initialisation of web elements using class Constructor and method pageFactory.initElements(driverRef, this);
* Then we do element utilization by providing public getter methods for each web element. Since there are no setters method for element, with this we achieve encapsulation providing restricted access to the elements
* In POM, we also define Business Libraries as methods which are generic to the application and utilizes the web element and perform actions on them

**4. Difference between POM and PageFactory**

Ans.

* POM stands for Page Object Model which is a java design pattern preferred by google to develop object repository. As per POM design pattern, we create separate classes for all the web pages of the application. The objects/elements of each page are declared, initialized and utilised in their respective POM classes.
* Page Factory is an extended design pattern of POM which has a method called as “initElements()” which is used to initialize all the web elements declared in the pom class with latest updated driver reference

**5. Difference between @findBy and driver.findElements()**

Ans.

* **driver.findElements()** is used to locate web element. “findElements” is a methods provided by WebDriver interface.
* driver.findElement() will return a reference to the web element and driver.findElements() return a list of web elements
* Using driver.findElements() can lead to StaleElementReferenceException.

Eg: driver.findElements(By.id(“id1”)).get(0).click();

* Here we have identified a web element using a driver reference. But when we try to perform click action of that identified element, the element can either be no longer attached to the DOM or the page got refreshed or the element reference may be old. In this situation, we get StaleElementReferenceException
* **@FindBy** is an annotation provided by Selenium to identify web element or list of web elements
* Usage: @FindBy(id=”id1”)

private webElement nameTxt;

* @FindBy will declare a webElement at compile time and this element will only be initialized at runtime using PageFactory.initElements(driver,this) when the object is created with latest updated driver reference.

Since while initializing the web element located by @FindBy the latest driver reference is used, it avoids StaleElementReferenceException

**6. Explain @FindBy, @FindAll, @FindBys**

Ans.

@FindBy:

* This annotation is used to locate webelement or list of webelements using one search criteria
* Usage: @FindBy(locator =” value”) private WebElement element;
* Eg: @FindBy(name=”username”) private WebElement element;

@FindAll:

* This annotation is used to locate webelement or list of webelements using multiple search criteria or locators
* This annotation applies OR operation between multiple locator strategies
* Usage: @FindAll({@FindBy(locator1=”value”), @FindBy(locator2=”values”)})
* Example: @FindAll({@FindBy(id=”val1”), @FindBy(name=”username”)})
* Here, auto-healing process is used, i.e, if the element is located using first locator it will return the web element. If not, it will try to locate the web element using the second locator and so on

@FindBys:

* This annotation is used to locate webelement or list of webelements using multiple search criteria or locators
* This annotation applies AND operation between multiple locator strategies
* Usage: @FindBys({@FindBy(locator1=”value”), @FindBy(locator2=”values”)})
* Example: @FindBys({@FindBy(id=”val1”), @FindBy(name=”username”)})
* Here, the element satisfying all the locator strategies provided will be returned.

**7. Develop POM Class for a login Page**

Ans.

package vTiger.ObjectRepository;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

import vTiger.GenericUtilities.WebDriverUtility;

public class LoginPage {

// Identify the web elements using @FindBy , @FindAll, @FindBys

// annotations

@FindBy(name = "user\_name")

private WebElement UserNameEdt;

@FindBy(name = "user\_password")

private WebElement PasswordEdt;

@FindBy(id = "submitButton")

private WebElement SubmitBtn;

// Create a constructor to initialize the variables/ web elements

public LoginPage(WebDriver driver) {

PageFactory.initElements(driver, this); // initializes web element using updated driver reference

}

// Provide getters to access the web elements

public WebElement getUserNameEdt() {

return UserNameEdt;

}

public WebElement getPasswordEdt() {

return PasswordEdt;

}

public WebElement getSubmitBtn() {

return SubmitBtn;

}

// Define Business library - Generic methods for this application

/\*\*

\* This method will login to the application

\*

\* @param USERNAME

\* @param PASSWORD

\*/

public void loginToApp(String USERNAME, String PASSWORD) {

UserNameEdt.sendKeys(USERNAME);

PasswordEdt.sendKeys(PASSWORD);

SubmitBtn.click();

}

}

**8. What is StaleElementReference Exception? How POM will handle it**

Ans.

**StaleElementReference** exception occurs in situation where we identify a web element using a driver reference. But when we try to perform any action of that identified element, the element is either no longer attached to the DOM or the page got refreshed or the element reference may be old. In this situation, we get StaleElementReferenceException.

POM avoids this exception. In POM we use @FindBy, @FindAll and @FindBys annotations to identify the elements. And these elements are initialized with updated latest driver reference when the object is created using PageFactory.initElements(driver,this)

9. What is Auto healing?

Ans.

Auto healing process is a process where if the element is not identified with one locator, @FindAll annotation by default will check for another locator.

During execution, if one locators fails to identify the particular element, @FindAll annotation will retry to identify the same element with different locator

* Usage: @FindAll({@FindBy(locator1=”value”), @FindBy(locator2=”values”)})
* Example: @FindAll({@FindBy(id=”val1”), @FindBy(name=”username”)})

**Scenario based**

**1. Agile process demands frequent UI changes, How do u handle this situation**

Ans. In Agile process we can accommodate frequent UI changes using Page Object Model (POM).

* POM stands for Page Object Model
* POM is a java design pattern preferred by google to develop object repository. As per POM design pattern, we create separate classes for all the web pages of the application. The objects/elements of each page are declared, initialized and utilised in their respective POM classes.
* In every POM class, first we do element declaration. In this, we identify each web element using @FindBy, @FindBys or @FindAll annotations such as @FindBy(locatorStrategy=”value”) and declare the webElement as private
* Then we perform element initialisation of web elements using class Constructor and method pageFactory.initElements(driverRef, this);
* Then we do element utilization by providing public getter methods for each web element.
* Following this process, we avoid creating repetitive hard coded elements/locators inside test scripts. Thus, whenever the UI changes are there, we can simply update the locators inside designated POM classes.

**2. Session Id for a page keeps on Changing when ever the page is refreshed, How do hanlde**

**This situation**

Ans. . If the session id of a page keeps on changing whenever the page is refreshed, we can get “StaleElementReferenceException” on trying to perform actions on any element located using stale/old web driver reference.

This situation is handled using PageFactory which is an extended design pattern of Page Object Model.

In POM we declared the web elements using @FindBy, @FindBys and @FindAll annotations.

These web elements are initialised at runtime with the updated driver reference (thus avoiding stale reference exception and accommodating change in session id)

To initialize with updated driver reference, we use PageFactory.initElements(driverRef,this) method whenever the object is created.

Usage:

**public** **class** LoginPage{

@FindBy(name = "user\_name")

**private** WebElement UserNameEdt;

@FindBy(name = "user\_password")

**private** WebElement PasswordEdt;

@FindBy(id = "submitButton")

**private** WebElement SubmitBtn;

**public** LoginPage(WebDriver driver) {

PageFactory.*initElements*(driver, **this**); // initializes web element using updated driver reference

}

**3. One of web pages consists of 5 drop downs, How do handle it**

Ans. We handle dropdown by using generic methods available in WebDriverUtility class to handle dropdowns.

*Generic Methods in WebDriverUtility:*

//Method 1: This method will select option from dropdown with matching value

public void handleDropDown(WebElement element, String value){

Select sel = new Select(element);

sel.selectByValue(value);

}

//Method 2: This method will select value by visible text

public void handleDropDown(WebElement element,String text){

Select sel = new Select(element);

sel.selectByVisibleText(text);

}

//Method 3: This method will select value by index

public void handleDropDown(WebElement element,int index){

Select sel = new Select(element);

sel.selectByIndex(index);

}

- So for handling each dropdown, we can identify its locator and create WebElement type object for it and then call any of the above method based on the selection criteria.

**4. How do u handle synchronization in Selenium.**

Ans.

Synchronization is a process to bring script execution time (which is usually faster) in sync with the application load time.

To handle synchronization in Selenium we create generic methods in WebDriverUtility class using following waits available:

1. **Implicit wait:**

* This is an intelligent wait which works on all the findElement() and findElements() statements
* This waits for a specific duration of time for every element in the web page to load
* It is dynamic – It polls every 0.5 seconds to check if the element in available in DOM, If yes, it breaks out of the wait and moves to next line of code. If no, it continues to check every 0.5sec until the duration expires. If element is not found after the time expires, it throws NoSuchElementException
* Usage:

/\*\*

\* This method will wait for 20 seconds for all elements to get loaded

\*

\* **@param** driver

\*/

**public** **void** waitForPageLoad(WebDriver driver) {

driver.manage().timeouts().implicitlyWait(Duration.*ofSeconds*(20));

}

1. **Explicit wait:**

* This is an intelligent wait which waits for a particular condition to be satisfied until the end of wait duration
* It is dynamic – It polls every 0.5 seconds to check if the condition is true. If yes, it breaks out of the wait and moves to next line of code. If no, it continues to check every 0.5sec until the duration expires. If the condition is not satisfied even after the time expires, it throws TimeOutException
* Usage:

/\*\*

\* This method will wait until a particular element becomes visible

\*

\* **@param** driver

\* **@param** element

\*/

**public** **void** waitForElementToBeVisible(WebDriver driver, WebElement element) {

WebDriverWait wait = **new** WebDriverWait(driver, Duration.*ofSeconds*(20));

wait.until(ExpectedConditions.*visibilityOf*(element));

}

/\*\*

\* This method will wait until a particular element becomes clickabe

\*

\* **@param** driver

\* **@param** element

\*/

**public** **void** waitForElementToBeClickabe(WebDriver driver, WebElement element) {

WebDriverWait wait = **new** WebDriverWait(driver, Duration.*ofSeconds*(20));

wait.until(ExpectedConditions.*elementToBeClickable*(element));

}

1. **Thread.sleep(int milliseconds)**

* Thread.sleep() is a method in java which suspends the execution until the time provided in milliseconds expires
* This is a hard wait and it completely stops the execution for the specified time
* Usage: Thread.sleep(2000) will stop the execution for 2sec

1. **Fluent Wait:**

* Fluent wait is an extended dynamic/intelligent wait similar to Explicit wait, in which we can also define the polling frequency explicitly, i.e., instead of default 0.5 sec polling period, we can define our custom polling frequency
* This wait allows us to ignore certain exceptions during runtime

1. **Custom wait:**

* We can also define methods to create our own customised wait conditions

**5. What are the total number of pom classes present in your framework.**

Ans. Total number of pom classes in our framework depends on the number of web pages in application. So Total number of pom classes = total number of web pages

**Questions - MAVEN:**

1. What is Maven explain Advantages?

Ans. Maven is a Build management and build testing tool.

* Developers use this tool to create build, test it and to deploy it to the testing server or any other environment.
* Maven accomplishes this using certain commands, like:

1. mvn package – converts the source code into executable format
2. mvn compile – tests the project for compilation issues
3. mvn test – makes sure that the build is successfully created or not
4. mvn install ; mvn deploy– build is installed into testing server or any other environments

* Maven is used by Automation testers to test the application with recent framework by executing the test scripts
* Automation test engineers uses mvn clean, mvn validate, mvn compile, mvn test which together is known as Maven Build Lifecycle

Advantages of Maven:

1. Maven can handle dependency jars. “mvn validate” checks for missing dependencies and automatically downloads them
2. Checks integration & compilation issue between the framework component which can occur due to multiple engineers making changes to the framework that might affect the entire build
3. Provides folder structure to the framework/project.
4. Allows the user to run test scripts outside Eclipse IDE via command lines
5. Supports Jenkins for CD/CI
6. Supports command line parameters
7. Supports Profiling using which we can execute any of the suite xml file at run-time through command line
8. Since POM.xml in maven projects acts as project configuration file with all the dependencies, making dependency or version changes for tools becomes easier and we do not have to download the tools from mvn repository explicitly every time.

**2. Explain Maven Folder Structure**

Ans.

Maven provides following folder structure:

1. src/main/java – In this folder, we store *GenericUtilities* and *ObjectRepository*
2. src/main/resources – In this folder, we store driver executable files like chromedriver.exe, geckodriver.exe
3. src/test/main –Here we create module wise packages and inside these packages we create test classes consisting of test scripts.
4. src/test/resources –In this folder we keep the test data. Eg: CommonData.properties , TestData.xlsx file
5. pom.xml – pom stands for Project Object Model which acts as Project configuration file where all the dependencies and plugins are kept.

**3. Explain the dependencies feature in Maven**

Ans. Dependencies feature allows automation test engineers to configure various tools like selenium, testing, apache poi in their project. Using dependencies feature we don’t have to download updated versions of these tools if needed. We can simply make changes to version in dependency and the required tool will be automatically download and attached to our project.

Dependencies or jars are downloaded from Global repository (http://mvnrepository.com) to Local repository

(C://user/name/.m2)

**4. Explain Build life Cycle**

Ans. Maven Build Life Cycle is a process made of following maven commands:

a. mvn clean: This command cleans or deletes all the reports from “target” folder of our project

b. mvn validate: This command checks if any required dependency is missing or not. If missed, then it will automatically download it

c. mvn compile: The command checks compilation issues between the old framework and new features

d. mvn test: This command will identify all the test classes whose class name ends with word “test” and execute all the test scripts in them.

**5. What are the different maven commands**

Ans. Maven commands used by Developers:

1. mvn package – converts the source code into executable format
2. mvn compile – tests the project for compilation issues
3. mvn test – makes sure that the build is successfully created or not
4. mvn install ; mvn deploy– build is installed into testing server or any other environments

Used by Automation test engineers:

1. mvn clean: This command cleans or deletes all the reports from “target” folder of our project
2. mvn validate: This command checks if any required dependency is missing or not. If missed, then it will automatically download it
3. mvn compile: The command checks compilation issues between the old framework and new features
4. mvn test: This command will identify all the test classes whose class name ends with word “test” and execute all the test scripts in them.

**6. What happends if we execute mvn test**

Ans.

* If there is no TestNG.xml file mentioned in pom.xml file, mvn test will work at class level and execute all the test classes whose name ends with “test”
* If we use surefire plugin, it will load the relevent testNG.xml file in <suiteXmlFile> and this

testNG.xml file is recognised and executed by maven with mvn test command.

**7. Write maven command to provide username and password through cmd line**

And. mvn -Dusername=sharun -Dpassword=test123 test

**8. Write maven command to execute only one test class through cmd Line**

Ans. mvn -Dtest=testClassName test

**9. Write maven command to execute only one test method inside a test class through cmd Line**

Ans. mvn -Dtest=testClassName#testMethodName test

10. How do you execute suite xml file through cmd line?

Ans. To execute suite xml file through cmd line, we need SUREFIRE plugin. Using this plugin in pomx.xml, we can mention the suite xml file we want to execute inside <suitexmlfile> tag.

SureFire plugin will load the relevant testNG.xml file in <suiteXmlFile> and this testNG.xml file is recognised and executed by maven with mvn test command.

11. I have 5 different suite xml files for 5 different execution, i want to decide which

suite file to be executed during runtime, what is your approach.

Ans. For this we use Maven Profiling. Maven Profiling allows us to create different profiles inside pom.xml such as smoke (to execute SmokeSuite.xml) , regression (to execute RegressionSuite.xml) , etc. and load different testing.xml files in each profile.

Eg:

<profiles>

<profile>

<id>Smoke</id>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.0.0-M8</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>testng\_SmokeSuite.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

</plugins>

</build>

</profile>

<profile>

<id>Regression</id>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.0.0-M8</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>testng\_RegressionSuite.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

</plugins>

</build>

</profile>

<profiles>

We can then execute any file using maven cmd as :

-> mvn test -P profileId --> here -P stands for profile

Eg: mvn test -P smoke

**12. how do you read data from cmd Line?**

Ans. To read data from cmd line, we Maven parameters like -Dkey=Value.

Eg: if we want to pass username and password through command line to test scripts we can use : mvn -Dusername=sharun -Dpassword=test1333 test

And in test classes we can read these values with System.getProperty(“username”) and System.getProperty(“password”)

**13. What are Maven parameters?**

Ans. Maven parameters allows us to provide run time parameters to our test scripts, ie, it is used to read data from cmd line during run time.

To read data from cmd line, we Maven parameters like -Dkey=Value.

Eg: if we want to pass username and password through command line to test scripts we can use : mvn -Dusername=sharun -Dpassword=test1333 test

And in test classes we can read these values with System.getProperty(“username”) and System.getProperty(“password”)

**14. What is Maven Profiling?**

Ans. Maven Profiling allows us to create different profiles inside pom.xml such as smoke (to execute SmokeSuite.xml) , regression (to execute RegressionSuite.xml) , etc. and load different testing.xml files in each profile. If we do not use maven profiling, we will have to update the pom.xml file with the suite file name we want to execute inside surefire plugin tag, which can corrupt our pom.xml file due to multiple changes.

For this purpose we use Maven profiling and create different profiles and load each profile for different suite xml file and then we can the required profile at runtime using maven command in cmd line

Eg:

<profiles>

<profile>

<id>Smoke</id>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.0.0-M8</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>testng\_SmokeSuite.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

</plugins>

</build>

</profile>

<profile>

<id>Regression</id>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.0.0-M8</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>testng\_RegressionSuite.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

</plugins>

</build>

</profile>

<profiles>

We can then execute any file using maven cmd as :

-> mvn test -P profileId --> here -P stands for profile

Eg: mvn test -P smoke

Questions - GIT:

**1. Why github?**

Ans. Github is a distributed cloud decentralized repository which can be accessed by anyone using internet.

It is a global repository and a technical platform used by developers to store the source code of application , automation testers to store automation framework, manual testers to store test cases or crs docs and devOps to store builds/build versions.

Github is used as a Source Code Management Tool (SCM) and as Version Control Tool (VCT)

Github is used because:

1. Since its cloud based repository so we do not need to maintain any software or hardware at our end
2. File Share between the team members becomes easier
3. It provides remote access thus can be accessed from anywhere via internet
4. Acts as a Version control system and maintains history for changes made by users & backup facility
5. GitHub also provide platform to review the code of automation test scripts created by different automation test engg. using Pull Request before making changes to main framework in Master branch
6. GitHub can also be used to handle conflicts before merging code to master
7. Github is mainly used by Jenkins to pick up latest framework to execute it

**2. What are different git commands**

Ans.

1. **Commit**:

Commit command will copy the framework from working directory (inside eclipse) to the local repository (.git folder generally created inside the project parent folder)

2. **Push:**

Push will copy the framework from local repository (.git folder) in our system to github global repository. To push code we need to provide global repo URL and user credentials: username and password/yoken

3**. Import + Clone URI:**

Import command will get the entire project or framework from global repository to our working directory, ie, to our local system.

Clone URI will copy the project from global repo to our local directory. It creates a folder with the same name as the framework URL in local system. This folder acts as local repository for further push actions.

4. **Pull:**

Pull will help us to get the changes pulled into our framework.

5. **Merge:**

Merge command is used to merge one branch at a time with the master branch.

6. **Fork:**

Fork command copies a repository from one github account to another github account.

7. **Rebase:**

This command will merge multiple ready to merge branched with master branch at once.

8. **Pull Request:**

It is an intimation to the manager that a new branch has been created and is ready to be merged into master branch. Later the manager can review the code and if there are no conflicts, he can merge the branch into the master branch

**3. What is branching in github**

Ans. To avoid unnecessary changes been made to the master copy of our framework which can lead to conflicts and affect the entire code, we use the concept of branching.

Using branching, we allow each test engineer working on test scripts to create a new branch and push their code in their own branch. Once they are done with their work, they can raise Pull Request to the manager to review and merge the changes to master branch.

Through this concept, master copy will be safe from unreviewed changes and thus avoid unnecessary conflicts and changes being made to master copy.

Every test engineer can only pull from master branch and only the manager/reviewer can make changes to master

**4. What is difference between git and github?**

Ans. Github:- Git Hub is a global repository which is used by engineers to maintain their source code in one place

2. Git (Git client):- It is a software installed in client system used to communicate with github.

A few common git client s are:

EGit: Egit is an Eclipse git plugin

GitDeskTop/GitBash: installed externally to execute git commands through cmd line

**5. What is pull request?**

Ans.

It is an intimation to the manager by automation tester that a new branch has been created by him and it is ready to be merged into master branch. Later the manager can review the code in the branch in github and if there are no conflicts, he can merge the branch into the master branch

**6. What is difference between merge and rebase?**

Ans.

Merge command is used to merge one branch at a time with the master branch.

Rebase command will merge multiple ready to merge branched with master branch at once.

**7. can v pull without import?**

Ans. No. We can only use pull after the project has been imported and thus is existing in our working directory.

**8. can v push without commit?**

Ans. No. We cannot directly push the changes from working directory to global repository. First we need to commit the changes from working directory to local directory. And then we can push from local repo to global repo.

**9. Explain github architecture**

Ans. **Github architecture has three stages as:**

* **Working directory**: This is created when a Git project is initialized onto our local machine and allows us to edit the source code copied.
* **Local repository**: If no further edits are required to be done, then we can go ahead and apply the **git commit** command. Git commit will copy the changes from our working directory into local repository (.git folder created in parent framework folder)
* **Global repository:** Global repo refers to github repository which is used to maintain our source code or framework at one place. We can pull the changes available in global repo to our local system using “git pull” and we can push the changes available in our local directory to global repository using “git push” and github repo url and user credentials