Advance Selenium:
Selenium and Experienced candidates only

Contents:

1. Locators - Dynamically changing elements
2. Handling dynamic WebTables
3. Synchronization - Fluent Wait & Custom Wait
4. JavascriptExecutor
   - scroll bar
   - click
   - pass the data to element
   - clear the data
   - refresh the page
5. Data Driven Testing - JDBC (Java DataBase Connectivity)
6. POM - Auto healing
7. TestNG - Group Execution, Listeners, @DataProvider
8. Report - Advanced reporting - Extent Reports
9. Maven
   - What is Maven?
   - Advantages
   - Maven build life cycle
   - Run test scripts using cmd prompt
10. Framework
    - Hybrid framework
    - BDD cucumber tool framework
11. GitHub
    - Git Architecture
    - Pull & Push
    - Branching
    - Merging
12. Jenkins
    - What is Jenkins?
    - Advantages
    - Types of execution
    - Build pipeline
    - Extract HTML reports in Jenkins
13. API Testing
    - SOA Architecture
    - PostMan
    - JSON
    - http request and http response

Locators:
Static methods of By class(By is abstract class)

1. id
2. name
3. classname
4. tagname
5. linktext
6. partiallinktext
7. css - tagname[AN='AV']
8. xpath -
   - xpath by attributes -> //tagname[@AN='AV'] --> and / or - Auto Healing
   - xpath by text() - //tagname[text()='TV']
     - //tagname[.='TV']
   - xpath by contains()
     - //tagname[contains(@AN,'AV')]
     - // tagname[contains(., 'TV')]
   - xpath by group index
     -(xpath expression)[position value]

- xpath by traversing
  - independent and dependent xpath
    > Forward traversing
    > Backward traversing
  - xpath by axes
    > parent
    > child
    > ancestor
    > descendant
    > sibling functions
      * following-sibling
      * preceding-sibling
    > preceding
    > following

Notepad : xpath expressions

5. Dynamic xpath/ xpath by traversing
   1) Independent and Dependent xpath
   2) xpath by axes

1) Independent and Dependent xpath :
   Independent xpath - static element xpath
   Dependent xpath - Dynamic element xpath

Steps to write dynamic xpath :
1. Identify static element and Write xpath
2. Identify common parent
3. Traverse to dynamic element from common parent

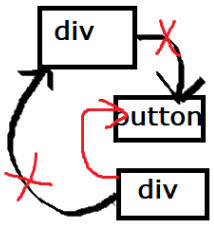
1. Forward Traversing :
   Traversing from parent to immediate child in html tree structure.
   We use '/'

Exercise 1:
Karthikeya2
Ek Villain Returns
Vikrant Rona

API docs of C#, Ruby, Java, Python, Javascript

2. Backward Traversing:

Navigating from child to immediate parent is called backward traversing
We use '/..'



1. Open the browser and enter demoapp.skillrary.com
   Mouse Hover on course,
   - traverse from Cucumber to course button
   - traverse from munit Testing to course button

2. Open the browser and enter demoapp.skillrary.com, Mouse hover on course and select Selenium Training, traverse from Add To Cart button to Se

Exercise 2 - Backward Traversing

1. Open the browser and enter https://www.bollymoviereviewz.com/2013/04/bollywood-box-office-collection-2013.html, traverse from disaster to Shamshera

2. Open the browser and enter selenium.dev, click on downloads and traverse from Changelog to C# image

3. Open the browser and enter selenium.dev, click on downloads and traverse from Changelog to Javascript image

2. xpath by axes :

In order to optimise xpath expressions we use xpath by axes

1. parent : To traverse to immediate parent
2. child : To traverse to immediate child
3. ancestor : To traverse to any parent
4. descendant : To traverse to any child
5. Sibling functions :
   1. preceding-sibling : Immediate sibling above given tag
   2. following-sibling : Immediate sibling below given tag
6. preceding :Any above tags
7. following : Any below tags

Syntax :
//tagname[]/axes::tagname

2. child

open the browser and enter demo.actitime.com, traverse from keep me logged in table to login button

3. ancestor:
open the browser and enter demo.vtiger.com, trverse from resouces table to vtiger logo

4.descendant:
open the browser enter flipkart.com, type watches in search TF, click on first watch, traverse from star ratings box to fAssured logo

parent, child, ancestor, descendant

Sibling functions:
1. preceding-sibling : Any node which is at same level which lies above the selected node

/preceding-sibling::

2. following-sibling : Any node which is at the same level which lies below the selected node

/following-sibling::

Scenario:
Open the browser
Enter ajio.com
Go to kids section
Select 0-2 years section
Fetch the price of third product
print product name and price in console

preceding : It selects all the above nodes which are at the same level of selected node though the parent is different

following : It selects all the below nodes which are at the same level of selected node though the parent is different

Open the browser, enter ajio.com
handle notification popup
go to kids section
select 0-2 yrs
fetch the product name and price of the third product and print in console

ctrl+shift+o

Scenario 2:
Open

MAVEN:
Maven is a build management tool/build testing tool which is used to check the integration issues between framework components whenever modification happens in the framework

Why Maven?
When multiple engineers are working on the same project, ther is a chance that if one engineer makes modification to the framework then it might effact entire project. In order to resolve these issues we go for Maven project

Advantages of Maven:
1. Handles dependency jar files
2. Integration issues between framework and components of the framework
3. Create framework configuration easily
4. Default folder structure
5. We can run test scripts in command line
6. Supports Jenkins

Dependencies to be added:
1. selenium-java
2. webdrivermanager
3. testNG
4. apache poi

Data Driven Testing:(Parameterization)

The process of fetching the data from external resources like properties file, excel and database and using it in test scripts is DDT.

We have two types of data :

1.Common data: url, login credentials, browser

2. Test data : test script specific data

Advantages of DDT:

* Maintenance of test data is easy
* Modification of test data is easy
* Run test scripts with multiple data
* Avoids hard coding
* data sharing is easy

1. Properties file
2. Excel file
3. JDBC(Java DataBase Connectivity)
4. xml file

Why DDT?

As per the rule of automation we shouldn't hard code the data in test scripts, also modifying data in test scripts is a tedious job instead we store the data in external resources and fetch it whenever required.

Steps to download and install mysql:

1. Open the browser and type 'mysql installer for windows'
2. Click on 'Download MYSQL Community Server' link
3. Click on 'Archives' tab
4. Select '5.5.62' version from the product version dropdown
5. Click on Download button against 'Windows (x86, 64-bit), MSI Installer'

Installation:

1. Open the downloaded .exe file
2. Next
3. Select 'I accept license agreement checkbox' and click on Next
4. Click on Complete button
5. Click on Install
6. Click on Yes button
7. Click on Finish button
8. Click on Yes button
9. Click on Next buttons 9 times until you find 'Enter the root password'
10. Enter the password as - root
    Confirm password - root
11. Click on Next
12. Click on Execute
13. Click on Finish

Check:

1. Go to search button and type mysql
2. Click on pin to taskbar (It will be added to taskbar)

Port Number : 3306
Username : root
Password : root
Url : jdbc:mysql://localhost:3306/
(open the browser and type 'mysql db url')

1. show databases;
   displays the list of databases present

2. create database database_Name;
   creates a new database with the given name

3. use database_Name;
   changes to particular database

4. show tables;
   displays existing tables

5. create table table_Name(col1_Name datatype(size) constraint, col2_Name datatype(size) constraint);
   creates a new table in the database

6. insert into table_Name(col1_Name,col2_Name,..) values('abc', 'xyz', 10,..);
   inserts the values into the table

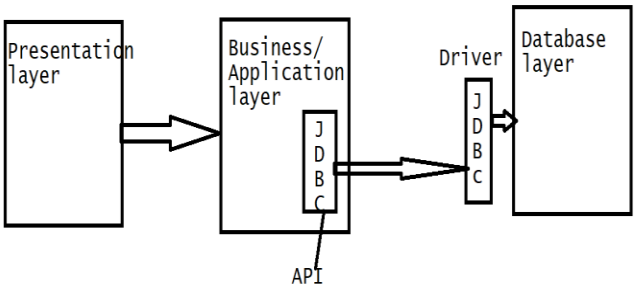
7. select * from table_Name;
   displays all the columns and data in th table

8. desc table_Name;

   displays table description

9. delete from table_Name where col_Name = value;
   deletes particular record in the table

JDBC(Java Data Base Connectivity)



JDBC is a J2EE(Java 2 Enterprise Edition) API which is used to connect to any database using java code.

JDBC API :It's an API which is available in java.sql package, used to communicate to database (DB independent)
It consists of interfaces and abstract methods

JDBC Driver : It is implementation of JDBC API, it is DB dependent
It is provided by database vendor

Types of Databases:

1. SQL Database :
   We store the data in the form of tables.
   Ex: MYSQL, SQL+, Oracle

2. NoSQL Database :
   The data is stored in JSON(Java Script Object Notation) format.
   Ex: NoSql, MongoDB, Cassandra

Why JDBC?
It is used by developers as well as Automation Engg

Developers - To validate data with respect to their source code

Automation Engg - To set pre conditions for testing an application

MYSQL:
We have two types of queries
1. Select Query : The query which has select clause in it.
   Ex: All DQL statements

2. Non-select query: The query which doesnot have select clause in it.
   Ex: DDL,DML statements

last() -> It is xpath function to fetch the last value in a table

To fetch the data
from database :
Prerequisite:
Database(mysql) should be installed
Get the url of jdbc -> jdbc:mysql://localhost:3306/advsel
Add mysql connector/J dependency to pom.xml
> Open the browser and type maven repository
> Click on the first link
> Type 'mysql connector/J' in the search field and click on search button
> Click on the first link
> Select '8.0.29' version
> Copy the dependency and paste it in pom.xml between dependencies tags
> Save it

Exercise:
Delete data from the database, print the result as well as print the database contents.

```java
package jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

import com.mysql.cj.jdbc.Driver;

public class InsertDataIntoDatabase {

    public static void main(String[] args) throws SQLException {

        //Step 1: Create an object for Driver
        Driver dbDriver = new Driver();

        //Step 2: Register dbDriver to JDBC
        DriverManager.registerDriver(dbDriver);


        //Step 3: Establish JDBC connection
        Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/advsel","root",
"root");

        //Step 4: Create Statement
        Statement statement = connection.createStatement();
        //Step 5: Execcute Query

              int result = statement.executeUpdate("insert into
     wcsm18(EmpId,EmpName,Address) values(1006, 'Fgh',
     'Goa');");
              System.out.println("Rows effected: "+result);

              //Step 6: Close database connection
              connection.close();
         }

     }
```

Exercise 2:
Write a java program to get only the first and the third column value from wcsm18 table of advsel database.

```java
package jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import com.mysql.cj.jdbc.Driver;

public class FetchDataFromDatabase {

    public static void main(String[] args) throws SQLException {

        // Step 1 : Create an object for db driver
        Driver dbDriver = new Driver();

        //Import Driver from com.mysql.cj.jdbc only
        // new Driver(); -> throws sql exception

        //Step 2: Register this dbDriver to JDBC
        DriverManager.registerDriver(dbDriver);


        //Step 3: Establish the JDBC connection
        //Import Connection, Statement and ResultSet from java.sql package only
        Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/advsel","root","root");

        //Step 4: Create the Statement
        Statement statement = connection.createStatement();

        //Step 5: Execute the Query
        ResultSet result = statement.executeQuery(" select * from wcsm18;");

        //Step 6: Fetch the data
        while(result.next()) {

    System.out.println(result.getInt("EmpId")+"\t"+result.getString("EmpName")+"\t"+result.getString("Address"));
        }

        //Step 7: Close database connection
        connection.close();
    }

}
```

Properties file:

It is a java feature file where in the data is stored in the form of key-value pair.

Why Properties file?

It is light-weight and faster to read data from properties file compared to any other files. Java has its own class to fetch and write data to properties file.
Usually all the common data is stored in properties file.

Scenario:
Store browser name, url and login credentials in properties file
Open the respective browser -> switch case
Enter the url of facebook.com
login to the facebook
get the title of page and verify if the home page is displayed or not
logout
Enter new credentials to properties file and login again

Disadvantages:
- Can't fetch multiple data
-Can't store the data in organised manner
-Since everything is stored in key-value pair searching for particular data is tedious job

Excel file:

Excel file is used to store test script specific data
Data will be stored in the form of table
We should integrate Selenium with apache poi
Apache poi libraries provide classes and methods to fetch data from and write data into Excel

Pre-requisite: Add dependency for apache poi-ooxml
We should have excel file added to our project

Synchronization:
The process of matching selenium speed with application speed is called synchronization.

1. Thread.sleep() -> blind wait, it is java wait statement
2. implicitlyWait -> It is Selenium wait statement which synchronizes findElement() and findElements() methods. It throws NoSuchElementException if element is not found.
Polling period is 500ms/0.5s
Syntax ->
Version 4
```
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
```
Version 3
```
driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
```

3. explicitlyWait -> It is also Selenium wait which synchronizes all the webdriver methods including findElement() and findElement(). Here we have to specify the condition. It throws TimeOutException if element not found. Polling period is 500ms/0.5s
Syntax ->
```
WebDriverWait wait = new WebDriverWait(driver,Duration.ofSeconds(10));
wait.until(ExpectedConditions.visibilityOf(element));
```

4. FluentWait
   It is a Selenium wait statement which makes it possible to change the polling period unlike implicit and explicit waits.

   Syntax -

```
FluentWait wait = new FluentWait(driver)
.withTimeout(timeout,SECONDS)
.pollingEvery(timeout,SECONDS)
.ignoring(Exception);
```

   Scenario:
   Open the browser
   Enter amazon.com
   Enter phone in search tab and click on search button

5. Custom Wait:
It is the wait customised by the user. User will write the logic for wait.

Debugging:

Checking the root cause of the error.
How to do debugging?
Run the program in debug mode
Debug mode -> running the program based on user instruction in slow motion

. -> Breakpoint - It pauses the execution at this point during execution

f5 -> Transfer the control from calling method to called method

f6 -> Transfer the control to next line

f7 -> Transfer the control from called method to calling method

ctrl+f2 -> To terminate the execution

f8 -> To resume the execution

Exercise:

WAP to print prime numbers between 1 to 20 and run it in debug mode

Scenario:
Open the browser
Enter the URL of demo.actitime.com
Enter Username as admin and password as manager
Click on login
Close the browser
WAP to execute the above script and run it in debug mode

JavascriptExecutor:
Is used to handle-->
1. Scrollbar
2. scroll to element using coordinates/ element address
3. Click
4. Refresh the browser
5. sendKeys
6. get Title
7. get URL
8. Disabled elements
9. Navigate to application

Scenario:
Open the browser
Enter amazon.com
Type dresses in search text field
Click on serach button
Scroll till desired element and click on the element

Generic Libraries:

It is one of the framework components which is common for all the test scripts.

It contains generic classes with reusable methods

Advantages:

1.Reusability of code
2.Test Script optimization
3.Code readability increases
4.Avoids duplicate codes
5.We need not remember syntax, just create and utilise it several times

src/main/java --> Create a package and name it as genericLibraries

Generic classes:

1. IConstantPath interface
   -> Property file path
   -> Excel file path
   -> Database URL

2. PropertyFileUtility class
   -> method to initialize property file
   -> method to fetch data from property file

3. ExcelUtility class
   -> method to initialize excel file
   -> method to fetch single data
   -> method to fetch multiple data
   -> method to fetch multiple data based on key
   -> method to write data into excel
   -> method to close excel

4. DatabaseUtility class
   -> method to initialize database
   -> method to fetch single data from database
   -> method to fetch multiple data from database
   -> method to write data into database
   -> method to close database

5. WebDriverUtility class
   -> Launch browser, maximize
   -> Navigate to application
   -> Implicitly wait
   -> Explicitly Wait
   -> Actions class methods
   -> Dropdown methods
   -> Javascriptexecutor
   -> Switching frames
   -> Switching windows
   -> Screenshot
   -> close browser

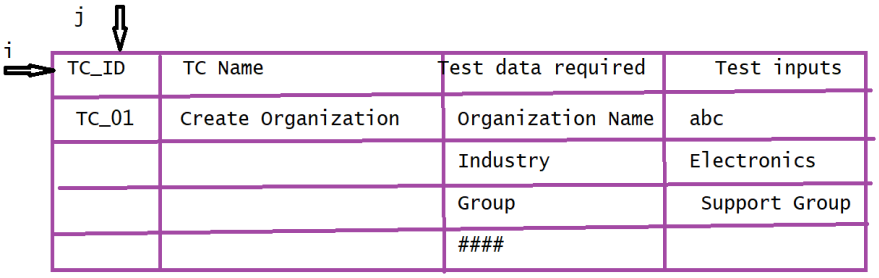
Uninstall old jdk

Install jdk 1.8

Add property to pom.xml
- Open the browser and type 'maven java compiler plugin'
- Click on 'Setting the -source and -target of the Java Compiler' link
- Copy the following

```
<properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
```

- paste it below </version> and above <dependencies>
- save pom.xml and update the project




```
Map<String,String> map = new HashMap<>();
for(int i=0 ; i<=sheet.getLastRowNum(); i++)
{
   if(df.formatCellValue(sheet.getRow(i).getCell(1)).equals("Create Organization"))
   {
       for(int j=i; j<= sheet.getLastRowNum(); j++)
       {
         map.put(df.formatCell(sheet.getRow(j).getCell(2)),df.formatCell(sheet.getRow(j).getCell(3));
          if(df.formatCell(sheet.getRow(j).getCell(2)).equals("####"))
               break;

       }
       break;
   }
}
return map;
```

To write status to excel:

```
for(int i =0; i<=sheet.getLastRowNum(); i++)
{
     if(df.formatCellValue(sheet.getRow(i).getCell(1)).equals(testName))
     {
          Cell cell = sheet.getRow(i).createCell(4);
          cell.setCellValue(data);
          break;
     }
}
FileOutputStream fos = new FileOutputStream(excelPath);
workbook.write(fos);
```

E:\srivalli\Selenium Batches\Selenium WCSE13\Attendance\Screenshot 2022-10-03.png

```
driver.findElement(By.name("imagename")).sendKeys("filePath");
```

POM(Page Object Model):

Object Repository:
It is the collection of all elements,locators and business libraries in one place.

We create Object repository in ->
src/main/java

Why Object Repository?
According to automation we should not hardcode any webelement because in Agile we have frequent GUI changes, maintenance and modification of elements becomes tedious. Oject repository provides a way to resolve the above problem

POM:
It is java design pattern recommended by google in order to develop object repository

POM has 3 stages:
1. Declaration
2. Initialization
3. Utilisation

1. Declaration:
Syntax - `@FindBy(LN="LV") private WebElement element_ref;`
`@FindBy(LN="LV") private List<WebElement> element;`

@FindAll ->
- It is one of the selenium annotation
- In order to identify an element with multiple locators
- Syntax :
  `@FindAll({@findBy(LN1="LV1"), @findBy(LN2="LV2"), ...})`
  `private WebElement element;`
- It works like OR operator
- Here we try to locate an element by checking the locators one by one. If the element is located then it returns element address immediately neglecting other locators in @FindAll.
- @FindAll supports 'AutoHealing' feature

AutoHealing: In automation, during execution if one locator fails to locate an element then it automatically checks for other locator to locate the element. This feature is called autohealing.

@FindBys -
- It is selenium annotation
- Syntax ->
  `@FindBys({@findBy(LN="LV"), @FindBy(LN="LV"),...})`
  `private WebElement/List<WebElement>  element_ref;`
- It works like AND operator
- Here we give multiple locators to find an element, it returns WebElement/List<webElement> only if all the @FindBy locators return WebElement/List<WebElement>

-> The return type of @FindBy, @FindBys and @FindAll is WebElement or List<WebElement>

2. Initialization:

We use constructor to initialise all the web elements in particular page

Syntax -
```
public PageName (WebDriver driver){

 PageFactory.initElements(driver,this);
}
```

StaleElementReferenceException :
- It is unchecked exception
- It occurs whenever we try to access an element with old address

Here during the initiallization the web elements are initialized to current page elements every time when the page is refreshed. Thus StaleElementReferenceException is handled.

3. Utilisation:
We provide business libraries or methods to perform actions on the web elements declared in that particular POM class.

Syntax -
```
public void methodName(){
  //actions;        }
```

Why POM?
It is well organized structured design pattern where we can maintain all the web elements page wise such that maintenance and modification of elements is easier.

Advantages of POM:
1. Handles StaleElementReferenceException
2. We can acheive encapsulation using POM
3. Maintenance and modification is easy
4. Improved code readability
5. Reusability of elements and business libraries
6. Since it is java design pattern we need not add any external libraries
7. Better fit for Agile
8. It supports AutoHealing

Steps to add getters and setters in POM pages:
1. Place the cursor in the class
2. Right Click
3. Go to Source
4. Go to Generate getters and setters
5. Select getter/setter/both depending on the requirement and click on Generate button

```
String path = "//a[.='%s']";

     WebElement
public    ' ' convertStringToDynamicXpath(String path, String replaceData)
{
   String requiredPath = String.format(path,replaceData);
   WebElement tab = driver.findElement(By.xpath(requiredPath));
   return tab;
}


public void clickRequiredTab(WebDriverUtility webdriver, String path, String replaceData)
{
   webdriver.convertStringToDynamicXpath(path,replaceData).click();
}


ENUM:

public enum TabNames {

ORGANIZATIONS("Organizations"),SIGNOUT("Sign Out");

private String tabName;

private TabNames(String tab){

this.tabName = tab;
}

public String getTabName(){
return tabName;
}
}
```

Robot class:

Robot class is used to perform all keyboard related
actions like ke press an key release.

Robot class should be imported from java.awt package

Usage:

```
Robot r = new Robot();
r.keyPress(KeyEvent.VK_TAB);
r.keyRelease(KeyEvent.VK_TAB);
```

Scenario:

Open the browser
Enter the url -> amazon.com
Right click on Today's deal
Open it in new tab

TestNG:

TestNG - Test Next Generation

It is unit testing tool used by developers and Automation engineers.

We have other unit testing frameworks:
Junit - java
NUnit - .net
Pydev - python
Jasmine/mocha - Javascript

TestNG is the combination of both JUnit and NUnit
It follows TDD (Test Driven Development) approach
It is developed as separate plugin to the IDE
It is a headless tool, IDE will give the UI for this testNG

Steps to add testNG plugin:

Open Eclipse
Go to Help
Go to MarketPlace
Type TestNG in find tab
Click on Install
Click on Install Anyway
Click on Restart Eclipse

Steps to add testNG to build path

Right click on project
Go to Build Path
Go to Add Library
Click testNG
Click on Next
Click on Finish

Steps to add testNG dependency:

Open the browser
Enter maven repository
Click on First Link
Type testNG in search tab
Click on first link
Copy the dependency and paste it in pom.xml of eclipse

Developers : Use testNG for Unit testing/ White Box testing

Automation Engg: Use testNG as Unit testing framework tool to write test scripts and execute them to test the application

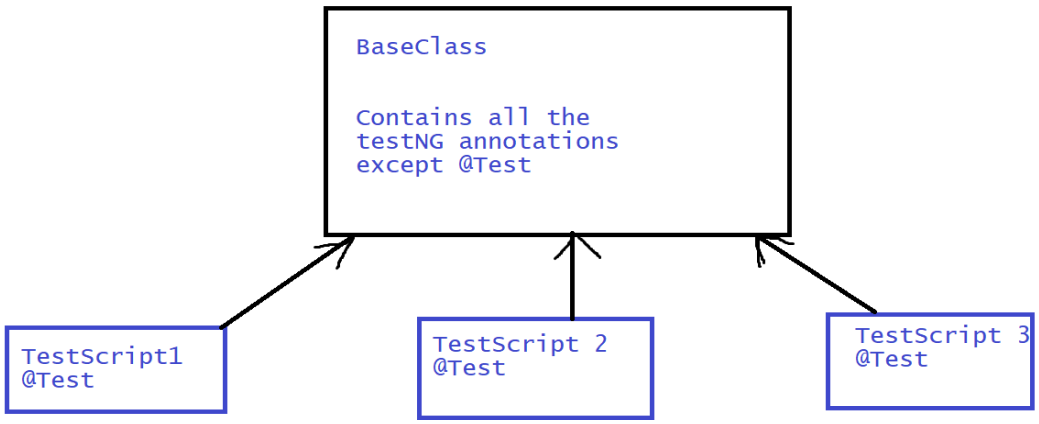
TestNG Annotations:
Annotations provide a way for controlled flow of execution

Annotations: java templates which provide meta data to JVM

Types of Annotations:
@BeforeSuite:It will be executed before <suite> tag in XML file.
It is executed only once
Ex:Open database connection
@BeforeTest: It will be executed before <test> in XML.
Ex: Initializing all utility classes
@BeforeClass: It will be executed before <class> in xml.
Ex: launching browser
@BeforeMethod: It will be executed before @Test method
Ex: Initialize all POMpages, navigate to application, login to application
@Test: It is responsible for the execution of all test scripts. It is like main method in java.

@AfterMethod: It will be executed after @Test method.
Logging out of application
@AfterClass: It will be executed after </class> in xml.
Closing the browser,Saving data to excel
@AfterTest : It will be executed after </test> in xml.
Close Excel
@AfterSuite: It is executed after </suite> in xml
Closing database connection

@DataProvider
@Listeners
@Parameters



BaseClass:

> All basic configurations
> It is developed in Generic Utility package in src/main/java
> All the test classes should extend to BaseClass

Priority:

```
@Test (priority=2)
public void firstTest(){
}

@Test (priority = 3)
public void secondTest(){
}

@Test (priority=1)
public void thirdTest(){
}
```

>We can set the priority to the test cases using a parameter priority
>Default priority is 0
> the lowest number will be executed first
> If priority is same to all the test cases, it will execute in alphabetical order of method names

Invocation Count:
It is used to run same test script multiple times

Usage: @Test(invocationCount = int)
Default Invocation count = 1

If Invocation Count = 0, it will not consider the execution of the test script

enabled = false :
It is used to disable the execution of test script

Usage: @Test(enabled = false)
By default enabled = true

dependsOnMethods :
It is used to make one @Test method depend on other

Usage: @Test(dependsOnMethods = "method_name")
       @Test(dependsOnMethods = {"method1", "method2",...})

Assertions:

Assertions are used in TestNG to validate the test scripts

In Automation it is mandatory to validate each and every test script

Normal if-else conditions are not recommended since if 'if' condition is satisfied 'if' block will be executed otherwise 'else' block will be executed. Hence if-else doesnot have the capability to fail the test script.

In testNG we have assertions feature which
> when actual result matches expected result it continues the execution
> when actual result doesnot match expected result it throws AssertionErrorException and also provides the line number where the failure occured.

We have two types of Assertions in testNG:
1. HardAssert/Assert
2. SoftAssert

1. HardAssert/Assert :

> Assert is the class
> Methods it has are all static methods -

```
Assert.assertTrue();
Assert.assertEquals();
Assert.assertFalse();
Assert.assertNotEquals();
Assert.assertNull();
Assert.assertNotNull();
Assert.fail();
```

> In Hard Assert, if assertion statement fails, it will stop the execution of the current block and throws AssertionError poviding the line number where the error occured and proceeds to next block skipping the execution of other statements of the current block.
> We use Hard Assert only for mandatory fields

2. Soft Assert:

> the class name is SoftAssert
> Methods in SoftAssert are non static methods

```
SoftAssert s = new SoftAssert();

s.assertTrue();
s.assertFalse();
s.assertEquals();
s.assertNotEquals();
s.assertNull();
s.assertNotNull();

s.assertAll(); -> should be given mandatorily at the end of
                  each block
```

> SoftAssert will not stop execution of current block even if the AssertionError occurs, instead it continues the execution and gives all the errors at the end along with the line numbers

> Hence we use SoftAssert for non mandatory fields

> assertAll() method should be given mandatorily at the end of each block else it will not execute remaining lines of the block if error occurs (behaves like Hard Assert).

@DataProvider:
To run the same test script multiple times with different set of data we use @DataProvider

-> Data is provided using two dimensional object array
-> Return type of @DataProvider is two dimensional object array
-> Each test script should have dedicated @DataProvider
-> We use @DataProvider in Data Driven Framework where we should test the application for multiple set of data like banking, ecommerce, booking applications.

Ex:

Bangalore 0,0	Hyderabad 0,1
Mysore 1,0	Bangalore 1,1
Bangalore 2,0	Kerala 2,1
Bangalore 3,0	Mumbai 3,1
Delhi 4,0	Kolkata 4,1

obj[5][2] ← Number of arguments in each set of data

↑ Number of sets of data

Batch Execution:

Collection of multiple test scripts is called batch. Executing multiple test scripts is called batch execution
It is done through xml file
Steps:
- Convert all the test scripts to testNG
- Run the xml file

We use batch execution for Full Regression testing

Group Execution:
-> Grouping the similar test scripts and executing is called group execution
-> To achieve group execution, each and every test script should be given group name in the annotations
-> If we don't specify the group name in any annotation it doesnot participate in the group execution

-> @BeforeTest(alwaysRun = true): It runs the particular annotation block irrespective of groups

-> We use group execution for Regional Regression testing

```
<suite name="Suite">
    <groups>
        <run>
            <exclude name="Smoke Tests" />
        </run>
    </groups>
    <test thread-count="5" name="Test">
        <classes>
            <class name="groupExecution.FifthTest" />
            <class name="groupExecution.ThirdTest" />
            <class name="groupExecution.FourthTest" />
        </classes>
    </test> <!-- Test -->
</suite> <!-- Suite -->
```

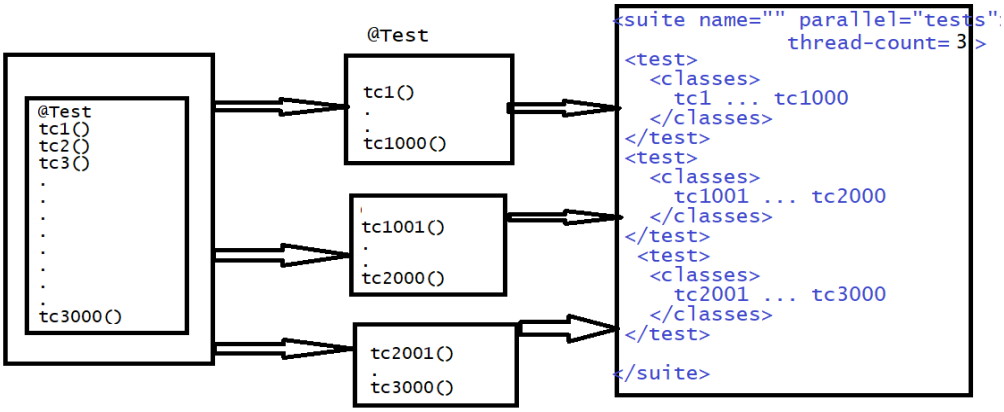
Unit Regression Testing:
Retesting the changed or modified feature is unit regression testing.

```
<suite name="Suite">
  <test thread-count=5 name="Test1">
    <classes>
       <class name=package.classname >
         <methods>
           <include  name=methodName  />
           <exclude  name=methodName />
         </methods>
       <class name=package.classname />
    </classes>
  </test>
</suite>
```

Parallel execution:
There are three types of parallel execution
1. Distributed parallel execution
2. Cross browser parallel execution
3. Cross platform parallel execution

1. Distributed parallel execution:
   Here we will distribute the test cases to different <test> (test runners)

Ex: Lets assume there are 3000 test cases
    1000 test cases takes 8 hrs
    Total test cases take 24 hrs



->We reduce suite execution time
-> We have to enable 'parallel = "tests"' and thread-count = 3 in suite tag to create multiple test runners
-> Default thread-count is 5
-> Thread count should be equal to number of test runners

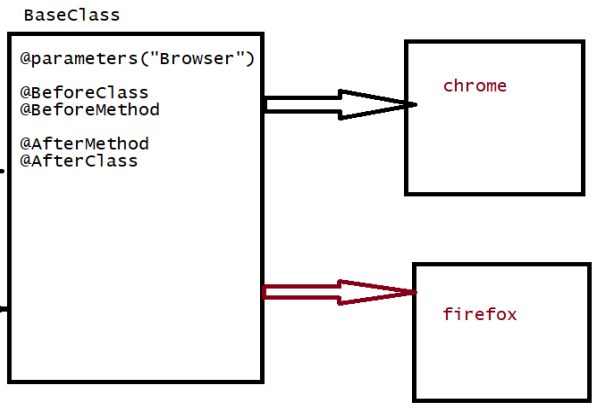
2. Cross Browser parallel execution/ Browser compatibility testing:
-> We execute same set of test cases in different browsers in parallel
-> To achieve cross browser parallel execution we should add <parameter> in XML and @Parameters for scripts
-> <parameter> is used to specify browser details

testNG.xml

```
<suite name="Suite" parallel="tests" thread-
count="2">
<test name="Test1">
 <parameter name="Browser" value="chrome"/>
  <classes>
     tc1 .. tc1000
  </classes>
</test>

<test name="Test2">
 <parameter name="Browser" value="firefox"/>
  <classes>
     tc1 .. tc1000
  </classes>
</test>

</suite>
```



Listeners:
It is a feature in TestNG which is used to capture runtime events during execution & perform appropriate action based on event type.

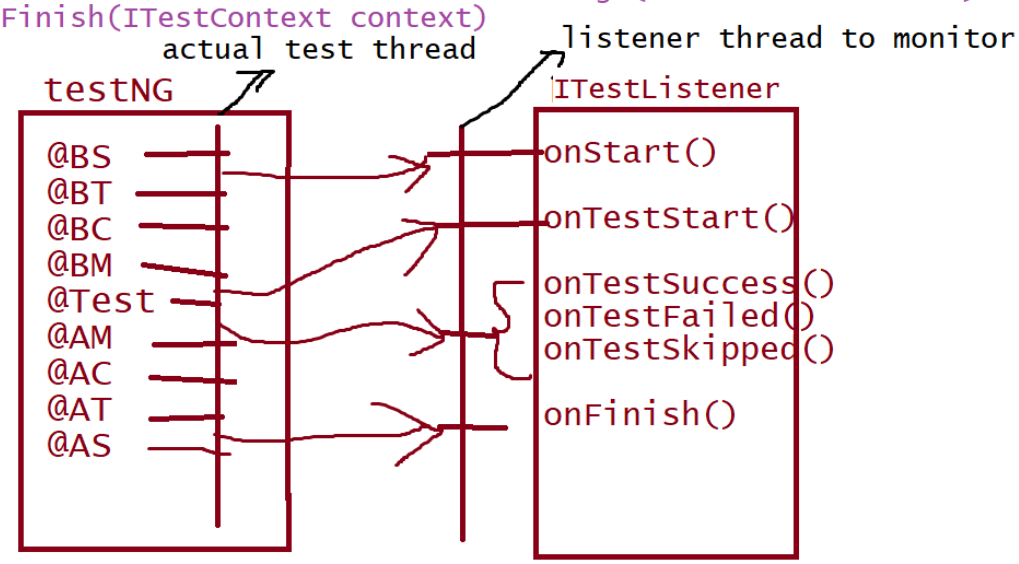
It monitors the test execution

Listeners have different interfaces :
ITestListener
ITestResult
IRetryAnalyzer

IAnnotationTransform

ITestListener:
It enables to capture events during runtime whenever test scripts fail or pass
We should provide implementation to this interface in Generic Utilities and enable listener annotation(@Listener) before every test
@Listener monitors the complete test execution

Methods in ITestListener interface:
onStart(ITestContext context)
onTestStart(ITestResult result)
OnTestSuccess(ITestResult result)
onTestSkipped(ITestResult result)
onTestFailure(ITestResult result)
OnTestFailedButWithinSuccessPercentage(ITestResult result)
onFinish(ITestContext context)



ITestResult interface:
It is used to capture pass/fail of test script during execution and it provides status accordingly. It is used as an argument for methods in Listeners.

IRetryAnalyzer interface:
- It is an interface which is used to re-run the failed test scripts.
- Failures of test scripts might occur due to synchronization issues or network issues, server issues. We usually fix these issues manually and try to re-reun these test scripts.
- To avoid this manual work we provide an implementation to IRetryAnalyzer such that it re-runs the test script automatically.

Step 1: Provide implementation to IRetryAnalyzer interface in Generic Utilities
Step 2: Use in @Test
Ex: @Test(retry Analyzer = qualified path of Retry Implementation)

IAnnotationTransformer :
It is an interface which has only one method used to transform one annotation to other

Whenever we have to re-run multiple test scripts we can provide Retry implementation in xml file as follows:
1. Convert retry implementation to listeners using IAnnotationTransformer.
2. Provide this converted retry implementation to <listener> in xml file

```
<suite name="Suite">
    <listeners>
        <listener class-name="genericLibraries.RetryToListenerConversion"></listener>
    </listeners>
    <test thread-count="5" name="Test">
        <classes>
            <class name="retryPractice.FirstTest" />
        </classes>
    </test> <!-- Test -->
</suite> <!-- Suite -->
```

Reports:
In TestNG, we have a feature which generates html report.

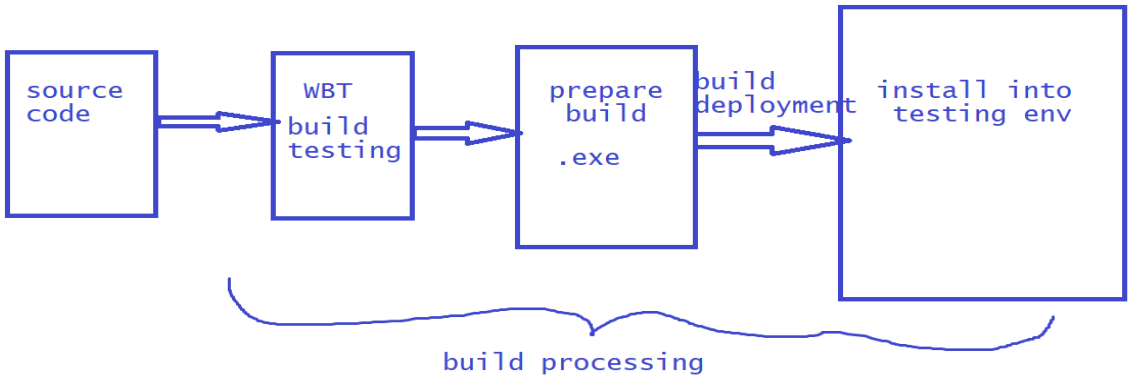
How to check html reports?
Go to test-output folder -> emailable-report.html
In this report we can find ->
passed,failed,skipped,execution time

Extent reports:
It is one of the reporting which is widely used in many organisations.
It provides graphical presentation of the reports
We can customise the report structure.

It has 3 major classes:
1. ExtentSparkReporter - It is used to set basic configuration for the report
2. ExtentReports - It is used to set the system information to the report
3. ExtentTest - It performs test script level analysis

Extent reports can be configured either in BaseClass or Listener implementation class of generic utilities package

Maven:
Maven is build Management tool/build process tool/build testing tool/build deployment tool
Maven has pom.xml file



Why maven in development?
In development, maven is used for build processes such as build testing, build creation and build deployment
Build testing : Testing compilation issues in the project and providing status of success or failure of the build
Build creation: converting source code to .exe file
Build deployment: installing build to testing environment

Why maven in Automation?
> to test automation framework build and test scripts
> When multiple engg work with same framework, chances are there if one engg makes modifications it might effect entire framework hence to avoid these issues we use maven

Advantages of Maven:
1. Checks integration between framework components
2. Handles all dependencies jars
3. Creates framework configuration quick setup
4. Provides framework folder structure
5. We can run testscripts on cmd line using maven commands
6. It supports Jenkins
7. It allows profiling
8. It also supports command line parameters

Maven Eclipse Plugin - available by default
Maven commandline plugin:
1. Open the browser and type 'maven download'
2. Click on the first link
3. Download apache maven 3.8.6 binary zip archive
4. Extract the files of downloaded archive

Maven Build Life Cycle:
mvn clean : to clean the older reports
mvn validate: to validate entire framework & downloads necessary jar files required to execute the test scripts
mvn compile: to check compilation issues in entire framework
mvn test: to execute all the test scripts
---

mvn package: to prepare the build
mvn install: to install the build
mvn deploy: to deploy the build
---

mvn test -Dtest=classname : execute particular test script

Surefire plugin: Used for execution of .xml files
Open the browser & enter 'Surefire plugin for maven'
Under 'Using Suite files' copy from <plugins> to </plugins> and paste it in pom.xml below version and above dependencies between <build> </build> tags

To pass the parameters in mvn cmd line:

mvn test -Dtest=classname -Dkey1=value1 -Dkey2=value2

```
In eclipse:  @Test
             public void demoTest(){
             String value1 = System.getProperty("key1");
             String value2 = System.getProperty("key2"); }
```

Profiling:

Profiling helps us to execute specific testng.xml via pom.xml file

Whenever framework contains multiple suite files or multiple .xml files in order to execute specific .xml file/suite file via cmd line we go for profiling.

```
<profiles>
        <profile>
            <id>regressionTests</id>
            <build>
                <plugins>
                    <plugin>
                        <groupId>org.apache.maven.plugins</groupId>
                        <artifactId>maven-surefire-plugin</artifactId>
                        <version>3.0.0-M7</version>
                        <configuration>
                            <suiteXmlFiles>
                                <suiteXmlFile>batchExecutiontestng.xml</suiteXmlFile>
                            </suiteXmlFiles>
                        </configuration>
                    </plugin>
                </plugins>
            </build>
        </profile>
        <profile>
</profiles>
```

mvn test -P regressionTests : command to run regressionTests

-P stands for profile

mvn -U clean validate install : To force update the project

GitHub:

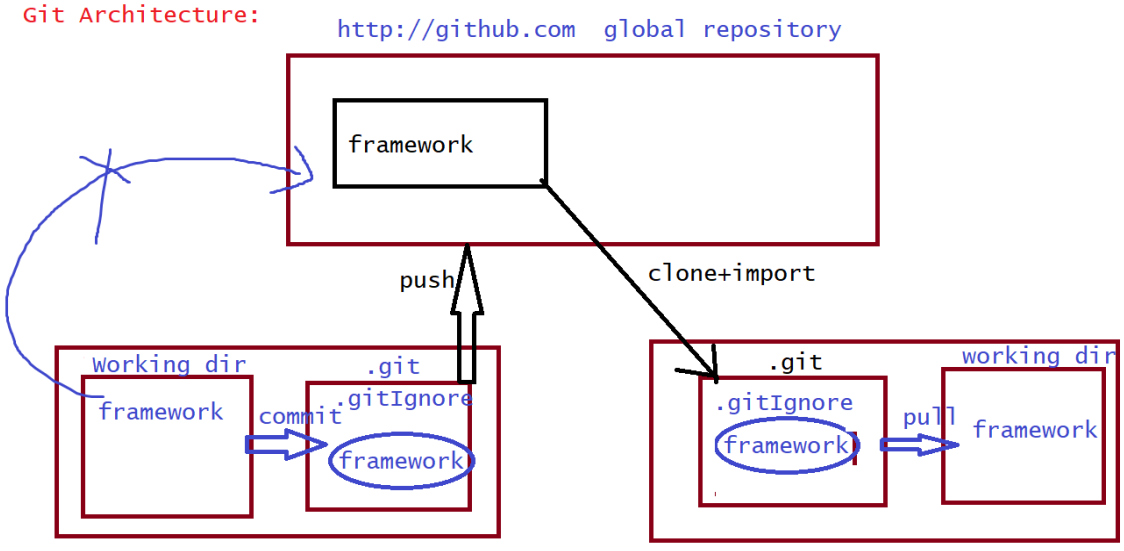
It is Source control tool/Code management tool/ Configure Management tool/Version control tool

What is GitHub?

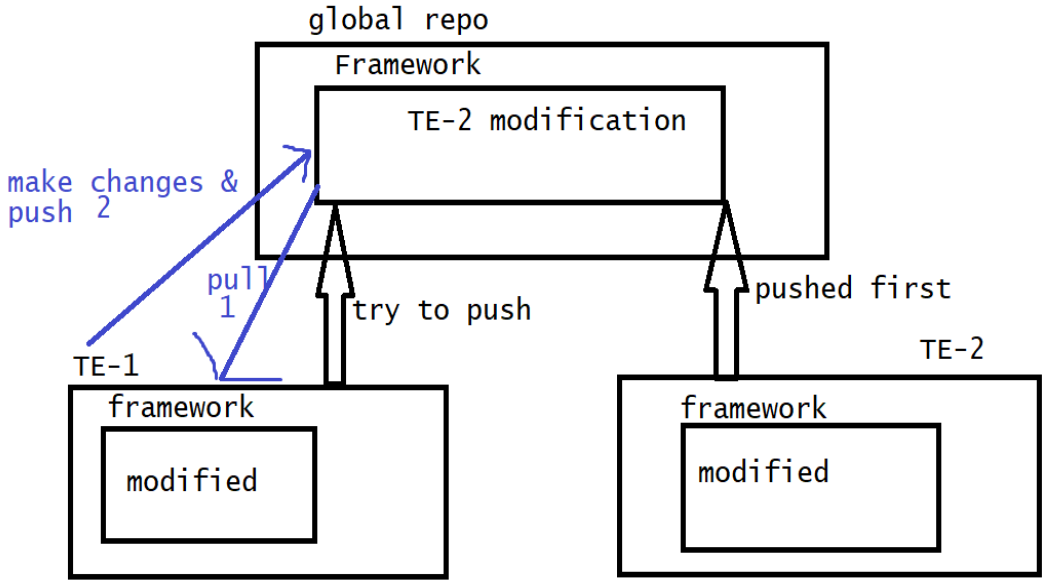
GitHub is a distributed cloud decentralized repository where we can maintain source code/Automation framework source code/CRS doc/build of the application in one place.

Developers Usage of Git: used to maintain the source code of the application
Automation Engg Usage of Git: used to maintain entire Automation framework
Devops Usage of Git: used to maintain build versions like .exe, .jar etc
Manual Test Engg Usage of Git: used to maintain entire CRS/test cases of the application

Git Architecture:



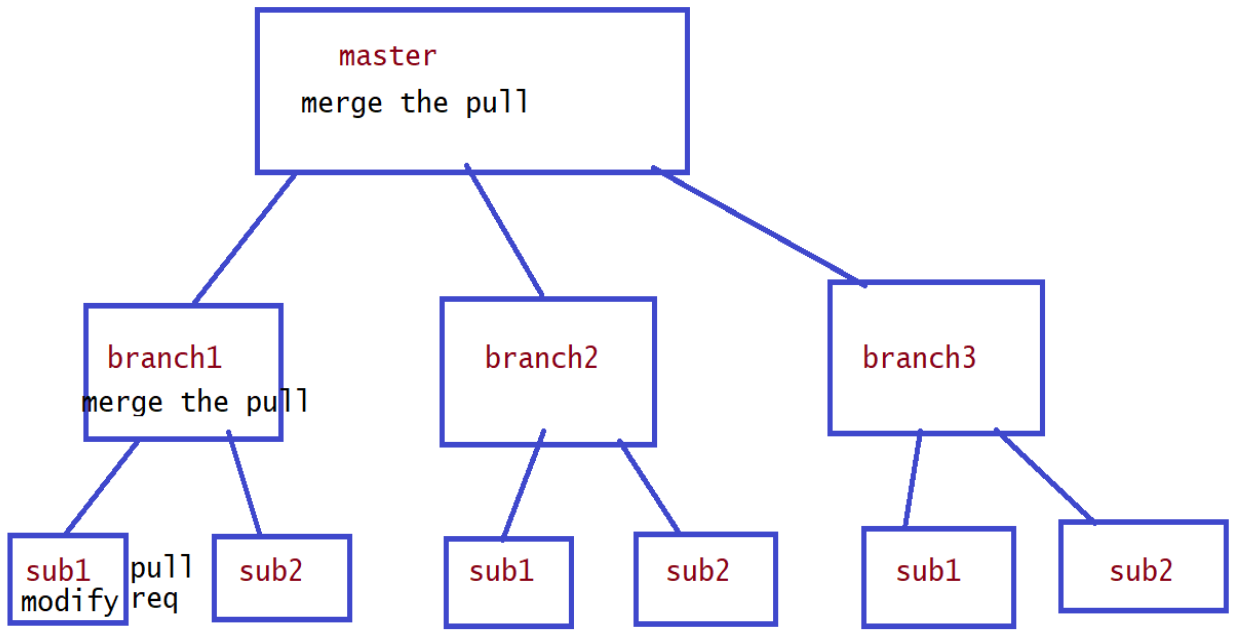
Git Conflict:



Whenever two or more engg modifies the framework and if engg 1 pushes the code first and engg 2 is also tries to push the code, then conflict arises that which change to be accepted and which one to be rejected. This is called Git conflict.

Solution: Engg 2 should first pull the code and decide which change to be retained and then make necessary changes and then push the code to global repository

Why branching? Global changes



JENKINS:
Jenkins is a CD/CI tool
CD - continuous Development
continuous Deployment
continuous Delivery

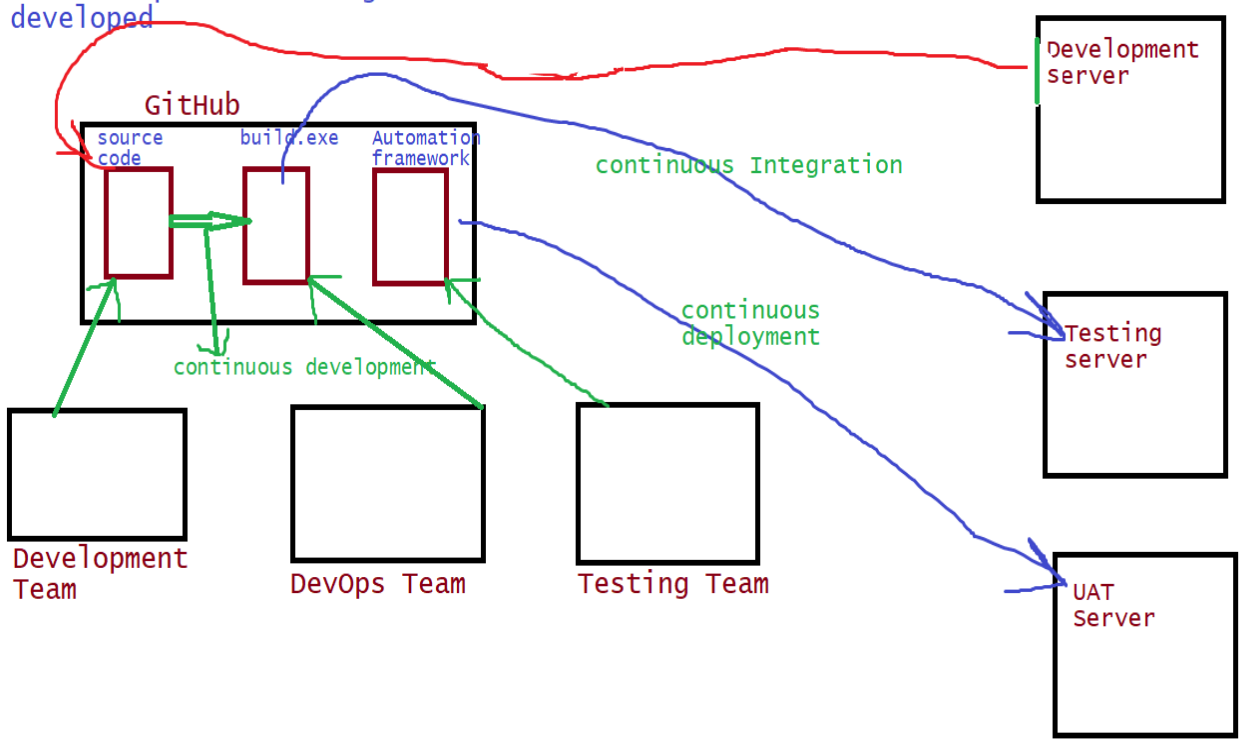
CI - Continuous Integration

Continuous Development: Continuously monitors git source code and creates a new build if any changes happened in git source code

Continuous Deployment: continuously gets the new build from git location and deploys the build to testing environment

Continuous Delivery: continuously gets the latest build from git and deploys in to UAT(User Acceptance Testing) environment

Continuous Integration: continuously triggers the execution of test scripts in testing environment whenever new build is developed



Integrations required
1. Maven Integration
2. Github Integration
3. Build Pipeline Installation
4. HTML Publisher

Set the paths from local repository
1. JDK path
2. MAVEN Path
3. Git path (.exe)

We have 3 stages of execution in Jenkins.
1. Ondemand: Whenever user hits 'Build Now', the build execution is triggered, this is called Ondemand execution

2. Scheduling: Here test engg schedules the time of execution such that it executes at that particular time only.
Format for Scheduling:
Min HH DOM Mon DOW
Min - minutes -> 19
HH - hours (24 hour format) -> 8
DOM - Day Of Month -> 10
Mon - Month -> 11
DOW - Day Of Week - 0 & 7: Sunday
1: Monday,2: Tuesday,3: Wednesday,4: Thursday,5: Friday,6: Saturday

Ex: 24 8 * * * -> executes every day at 8:24AM

3. Poll SCM(Source Code Management): It triggers the execution whenever new commit is made by test engg.

Framework:
It is a well organized structure of components.
It is the collection of reusable components and methods that facilitates development, modification, maintenance and execution of test scripts easier and faster.

Advantages of framework:
1. It is well organized
2. Stores data in external resources
3. Using testNG has lot of advantages
4. Using POM for object repository
5. pom.xml takes care of dependencies
6. Reusability,maintenance and modification is easy
7. Code optimization is acheived
8. Easy debugging
9. Re-run failed test scripts
10. Takes screenshot of failed test scripts
11. Test Script development is faster
12. Report generation

Disadvantages of framework:
Coding knowledge is required
Knowledge of multiple open source tools is required

Framework approaches:
1. TDD(TestDriven Development) approach:
Develop test scripts for the test cases provided by Manual test engineers. Automation engg develop various methods to write a test script and coding knowledge is required. Only technical person can understand this framework.

2. BDD(Behaviour Driven Development) approach:
Using layman language we develop test cases in feature files and we create step definitions and runner classes to execute the test script. This is understood by any person. Test cases and test scripts for respective test cases are developed by Automation engineer.

Layman language - Gherkin language

Types of frameworks:
1. Data Driven Framework:
The applications which has multiple data to be tested are developed using DDF. We use @DataProvider
Ex: Ecommerce, Banking, CRM, Booking

2. Method Driven Framework:
The application which has more reusable actions or features is developed using Method Driven Framework.
It is not suited for web application
This framework contains generic libraries and business libraries

3. Keyword Driven Framework:
If the framework is controlled by the keys given in the external resources then it is Keyword Driven Framework.
It is not suited for web application
For small applications we use Keyword Driven Framework
When Manual test engg is involved in automation Keyword Driven Framework is used.

4. Modular Driven Framework:
If an application has n number of modules and huge then we go for Modular Driven Framework

5. Hybrid Framework:
If the framework is the combination of two or more frameworks then it is called Hybrid framework

The possible combinations are:
1. Data Driven + Modular Driven
2. Data Driven + Modular Driven + Method Driven
3. Data Driven + Method Driven
4. Data Driven + Keyword Driven
5. Keyword Driven + Method Driven

Ours is a Hybrid Framework, combination of Data Driven, Modular Driven and Method Driven frameworks.

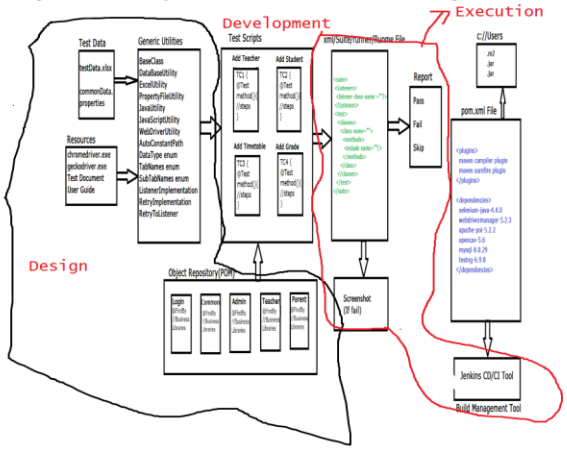
Components of our framework:
1. Test Data
2. Common Data -> src/test/resources
3. Generic Utilities
4. Object Repositories(POM) -> src/main/java
5. Resources(.exe files) -> Test Documents, Framework user guide - src/main/resources
6. Test Scripts - src/test/java
7. Reports
8. Suite files/Runner files/Runme file/xml file
9. Screenshot
10. GitHub - Cloud Repository
11. Jenkins - CD/CI tool
12. pom.xml - project object model - to handle versions and dependencies and plugins

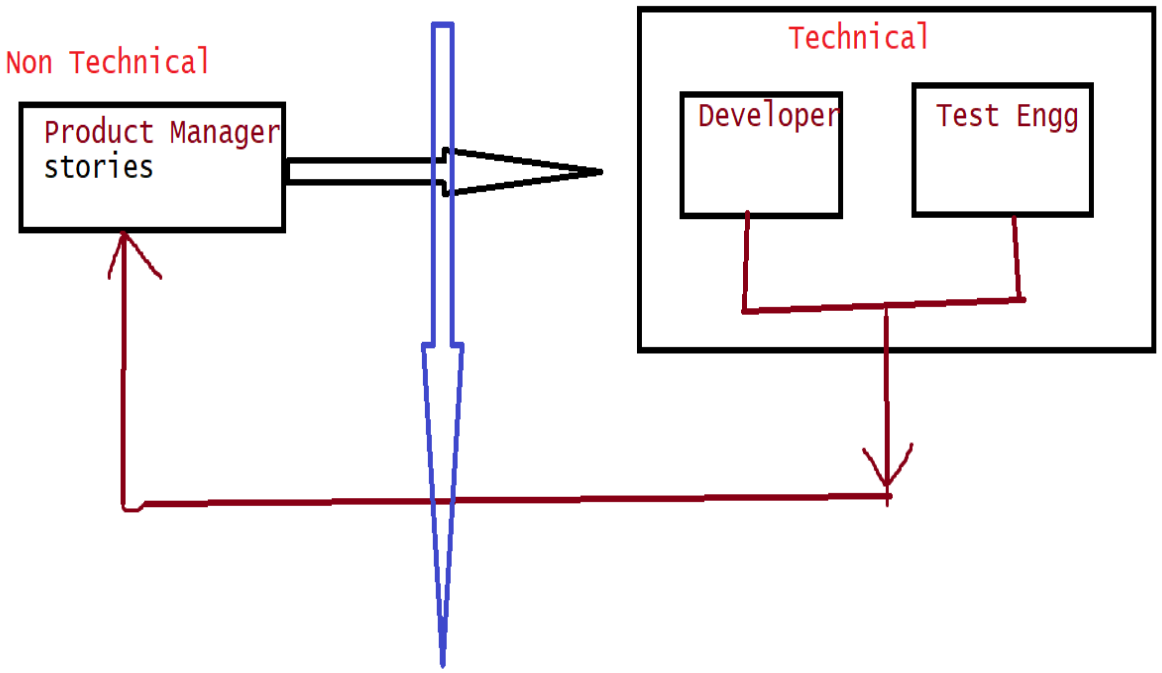
Exercise:
Prepare Framework User Guide
1. Framework Definition
2. List of components
3. Framework Architecture
4. Brief explanation of components, methods of generic utilities
5. Advantages of Generic Utilities
6. Explanation and Advantages of POM
7. Tools used in your framework
8. Advantages of framework
9. Explain your framework

Framework Architecture:
We have three stages in framework:
1. Design: Test data template, Resources, generic libraries and partially POM is developed here.
This job is done by Architect/Senior developer/Senior Test Engineer.

2. Development: Test Scripts are developed using Object Repository and generic libraries. Test data and POM pages are also updated here.
This job is done by Freshers/ Junior test engineers.

3. Execution: Creating xml files, conducting batch/parallel/group executions in local/jenkins/virtual machine/cloud computing and generation of reports is done here.
This job is done by Test Lead or Senior Test Engineer.





fills the gap between technical and non technical people in organization

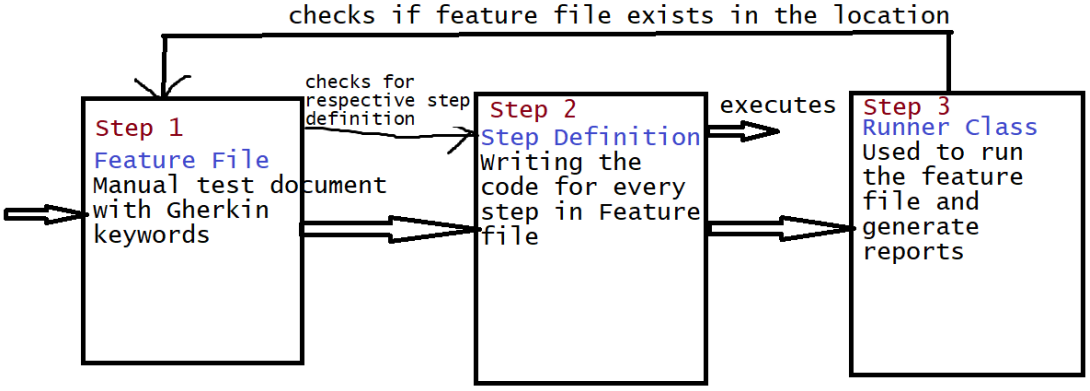
Cucumber Tool:
Open source tool which is used to write test cases using feature files and create step definitions for the same and execute and generate reports.

Pre-requisites to develop cucumber framework:
1. Create Maven project
2. Add following dependencies:
   - cucumber java
   - cucumber junit
   - JUnit
   - selenium java
   - webdrivermanager

Feature file: It acts like manual test case document to automate the test scripts. It is developed using Gherkin language.

Gherkin language: It is normal English language with set of keywords which provides a structure and meaning to develop feature file and program it.
Note: Gherkin keywords always start with Uppercase letters
Keywords in Gherkin Language:
Feature: To provide high level description of feature to be tested
Scenario: Describes the Scenario
Given: Describes initial content/pre condition of feature
When: Describes the action
Then: Describes the expected outcome of above action
And: Connecting keyword which can be used with Given, When, Then if there are multiple pre-conditions/actions/outcomes respectively

Cucumber WorkFlow:



Selenium Grid:
It is an open source tool in Selenium Community which is used to facilitate remote execution and compatibility testing.
It is collection of libraries from
Selenium RC + Selenium WebDriver + Selenium Grid Server

Why Selenium Grid?
Used to perform
- Remote Execution
- Cross Browser testing
- Cross Platform testing

Remote Execution:
Executing same test script on any other remote devices like computers on same network, cloud computing

How to perform Remote Execution?
To perform Remote Execution we make use of:

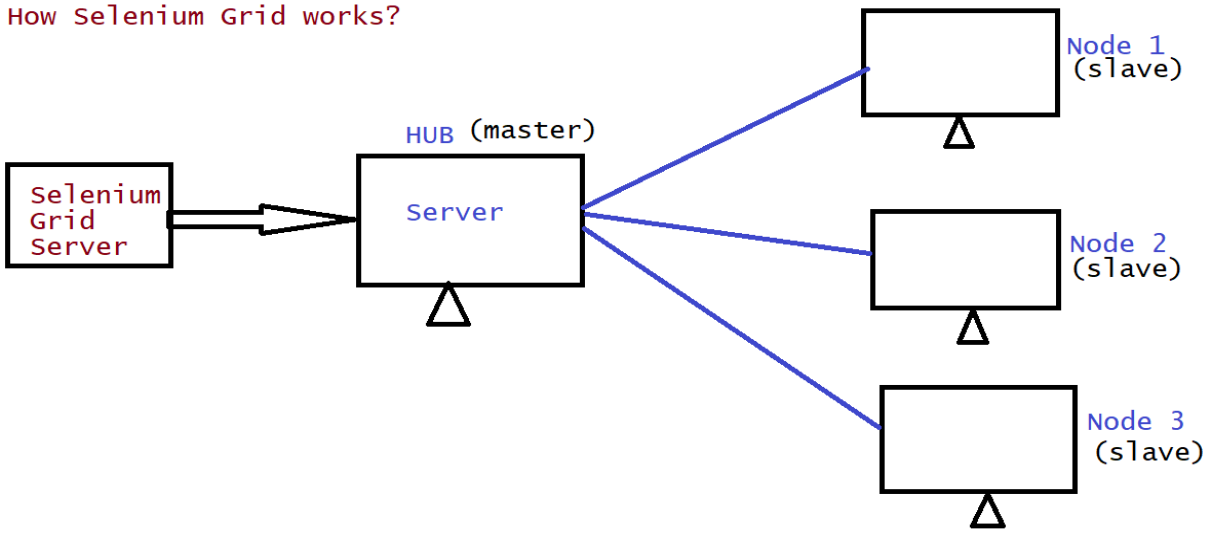
1. RemoteWebDriver: It is the implementing class of WebDriver interface which is used to perform remote execution using Selenium Grid. It is available in org.openqa.selenium.remote.RemoteWebDriver

2. DesiredCapabilities: It is a class which is used to set or change the capabilities of webdriver. During remote execution we should set the capabilities like browser name, browser version, platform to execute the test script. It is available in org.openqa.selenium.remote.DesiredCapabilities

3. URL: It is java class which helps to store remote address for remote execution. It is available in java.net.URL

Pre-requisites:
1. selenium standalone server
2. respective driver executables
3. IDE
4. jdk
5. selenium server

How Selenium Grid works?



HUB: It acts like a master. It receives command from selenium and bypass that to nodes. Default port number is 4444. If the port is busy we can customize it.

Node: It acts like slave. It receives command from Hub and executes the request.

Maximum 5 nodes can be connected to a hub.

1. Download selenium standaloneserver
2. Run the server as Hub role
   1. Open cmd
   2. To register Hub:
      java -jar <draganddropseleniumstandaloneserverjar> -port 9999 -role hub
   3. To register node:
      java -Dwebdriver.chrome.driver=<pathOfChromeDriver.exe> -jar
<draganddropseleniumstandaloneserverjar> -role node -port 7777 -hub <URL>
   4. Write the program using RemoteWebDriver, URL and DesiredCapabilities
and Run the program. Test Execution happens in node.