

Contents

- 1> Locators
 - Dynamic Changing Elements
 - Handling dynamic web tables
- 2> Synchronization
 - FluentWait
 - CustomWait
- 3> JavascriptExecutor
 - 1> scrollbars
 - 2> click
 - 3> pass data to webelement
 - 4> clear data
- 4> Data Driven Testing → JDBC
- ⇒ Assigning project

- 8> Framework
 - Hybrid Framework
 - Framework Architecture
 - BDD Cucumber Framework

9> Git & Hub

- Git Architecture
- Pull & Push
- Branching
- Merging

10> Jenkins

- what, Adv
- Types of Execution in Jenkins
- Build Pipeline
- HTML Publisher

5> POM

- @FindAll
- @FindBy
- AutoHealing feature

6> TestNG

- Group Execution
- Parallel Execution
 - Distributed Parallel
 - Cross Browser parallel
- Listeners
- @DataProvider

7> Maven

- what, Adv
- Maven Build Life Cycle
- Run testscripts on cmd Line

11> Selenium Grid

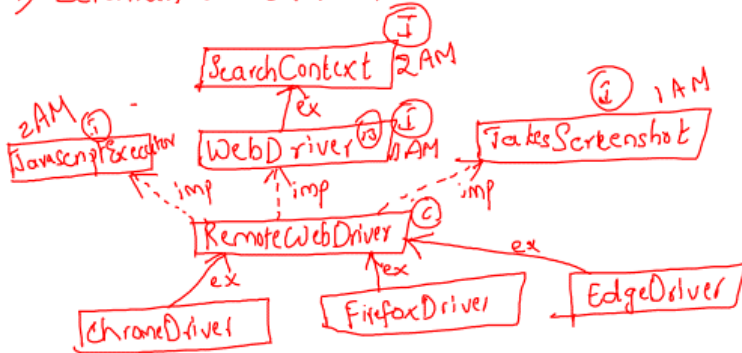
API Testing

- 1> what is API testing?
- 2> SOA
- 3> JSON
- 4> http request & http response
- 5> PostMan
- 6> CRUD operations
- 7> Authentication

→

Revision of concepts in Basic Selenium

➤ Selenium WebDriver Architecture



2> Navigation APIs

→ `back()` → `driver.navigate().back();` `navigate().to(url);`
→ `forward()` → " " `.forward();`
→ `refresh()` → " " `.refresh();`

`maximize` → `driver.manage().window().maximize();` →

`minimize` → " " " `.minimize();`

`fullscreen` → " " " `.fullscreen();`

Locators:

id, name, classname, tagname →

linkText, partialLinkText → links

css selector → `tagname[Attr = 'AV']`

xpath → Absolute → root to the element (/)

→ Relative → any parent to the element (//)

1> xpath by attributes - `//tagname[@Attr = 'AV']`

2> xpath by text - `//tagname[text() = 'TV']`

`//tagname[. = 'TV']`

3> xpath by contains - `//tagname[contains(@Attr, 'AV')]`

`//tagname[contains(text(), 'TV')]`

4) xpath by traversing

I 1) Forward & 2) Backward

II 1) xpath by axes

step 1: Identify ^{static element} ~~common parent~~ & write xpath

step 2: Identify common parent

step 3: Write tagname & index of dynamic element

5) xpath by groupIndex

(xpath Expression)[Position_Value]

Synchronization

1) implicit wait → 0-5 sec
2) Explicit wait } polling period

WebElement (i)

Action
click()
submit()
sendKeys()
clear()

getters
getText()
getSize()
getLocation()
getAttribute()

Validation
isEnabled()
isSelected()
isDisplayed()

Synchronization

- 1) implicit wait → } 0-5 sec
- 2) Explicit wait } polling period

WebElement ①

<u>Action</u>	<u>getters</u>	<u>Validation</u>
click()	getText()	isEnabled()
submit()	getSize()	isSelected()
sendKeys()	getLocation()	isDisplayed()
clear()	getAttribute()	

Handling webElements

- 1) Auto Suggestions → findElements()
- 2) Actions ②
 - mouse hover → moveToElement()
 - double click → doubleClick()
 - right click → contextClick()
 - drag and drop → dragAndDrop()
- 3) Dropdown - Select ③
 - selectByIndex()
 - selectByValue()
 - selectByVisibleText()
 - isMultiple()
 - deselectByIndex()
 - deselectByValue()
 - deselectByVisibleText()
 - deselectAll()

} perform() / build()

- 4> Scroll Bar → JavaScriptExecutor ①
- 5> Screenshot → TakesScreenshot ① → apache commons io
- 6> Popups → Alert → alert() < dismiss()
accept()
 - Hidden Division → findElement()
 - Notification → Browser specific
 - File Upload → AutoIT, sendKeys
 - Child Browser → getWindowHandler()
getWindowHandler()
- 7> Frames → driver.switchTo().frame(index)
- 8> Windows → " " • window() ;

1> Data Driven Testing -

1> Properties file →

2> Excel file → apache POI

2> POM - Page Object Model

3> TestNG - Test Next Generation

Notes → ~~Not~~ Monday
GitHub Link }

Saturday → WhatsApp group

Locators :

Locators are static methods of By class,
By is an abstract class.

Why?

In order to locate an element on web page

Types of locators :

- 1) id → when we have an attribute as 'id'
- 2) name → when we have 'name' as attribute
- 3) linkText → when we have text on a link
- 4) partialLinkText → when text on the link is lengthy / text contains partially changing values

- 5) classname } only when we identify
- 6) tagname } list of web elements

7) CSS Selector →
tagname [attr = 'av']
It is faster
does not support text
attributes are mandatory
Reverse traversing can't be
performed
1 of 1 matching is mandatory

- 8) xpath
 - Absolute xpath: root to immediate child using '/', length,
 - Relative xpath: any parent to child using
 - Basic relative '/'
 - Advanced relative →

Xpath Expressions → Basic Relative Xpaths

1) xpath by attributes —

//tagname[@AN = 'AV']

✓ → You can pass multiple attributes using 'and' and 'or'

//tagname[@AN1 = 'AV1' and @AN2 = 'AV2']

//tagname[@AN1 = 'AV1' or @AN2 = 'AV2']

Drawback —
Attributes are mandatory

Task 1 → Conduct a gmeet chat icon
Write xpath for that icon

Xpath → create a notepad
Task & Soln - xpath
↓

2) xpath by text() →

//tagname[text() = 'tv']

//tagname[* = 'tv']

text is mandatory
text can be lengthy

3) xpath by contains()

→ Handles lengthy text ✓

→ Handles spaces ✓

→ Handles partially changing elements ✓

→ Handles non-breakable spaces.

//tagname[contains(@AN, 'AV')]

//tagname[contains(text(), 'tv')]

//tagname[@AN = 'AV' and contains(-)]

Developer gives spaces in the text on the element using a command \rightarrow

\downarrow
xpath by text() X
xpath by contains() ✓

Task 2 \rightarrow Write xpath for (nbsp)
Products, Features, Resources in
demo.vtiger.com

1 \rightarrow Write xpath for 6 movie names in
bollywood review page (any 5) ✓

Advanced Relative Xpath

i) xpath by groupIndex ✓ X

Syntax - (xpath Expression)[Position Value]

GUI \rightarrow changes frequently [1] [2]
[100]

Task \rightarrow Write xpath for
Features, Products, Resources, Solutions
in demo.vtiger.com

2) xpath by traversing / Dynamic xpath:

① Independent & Dependent xpath

② xpath by axes

step 1: Identify static element
step 2: Identify common parent
step 3: Tagname & position of dynamic element

① Independent & Dependent xpath

→ Independent xpath: xpath of an static element

→ Dependent xpath: xpath of dynamic element

① Forward Traversing: Traversing from parent to immediate child using '/'

Task: selenium.dev → Downloads
Traverse from c# to API docs

② Backward Traversing: Traversing from an element to its immediate parent using './'


Task → demoapp.skillrary.com
course
Traverse from cucumber to course

Task:

1) bollywoodmovie review

Traverse from ~~box~~ BoxOfficecollection to movie name (5)

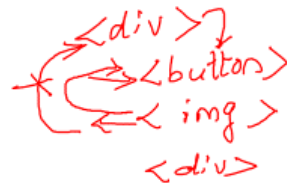
2) selenium.dev → downloads → changelog
to Javascript 'image'

3) demoapp.skillrary.com → course → Selenium Training →  image to 'Add to Cart'

2) xpath by axes: To optimise xpath expressions

1) parent → To traverse to immediate parent

Usage: `/parent::` → `'/..'`



Task → demoapp.skillrary.com → course

→ selenium Training

Traverse from AddToCart → `Se✓`

using parent axes

2) child → To traverse to immediate child

Usage → `/child::` → `'/'`

Task →

3) Ancestor: To traverse to any parent

Usage → `/ancestor::`

Task → demo.vtiger.com

traverse from resources to vtiger logo

4) descendant: To traverse to any child

Usage → `/descendant::`

3) Ancestor: To traverse to any parent

Usage → `/ancestor::`

Task → `demo.vtiger.com`

traverse from resources to vtiger logo

4) descendant: To traverse to any child

Usage → `/descendant::`

Task — Open the browser, type `ajio.com`,
Goto KIDS , 0-2 yrs , fetch the
product name & its price and
print them in console

6) preceding: To traverse from the given node
to the nodes at the same level above
having different parents

Usage → `/preceding::`

7) following: To traverse from the given node to
the nodes at the same level below having
different parents.

Usage → `/following::`

Test Scenarios

I Create Organization Test

- 1) Open the browser and enter url localhost:8888
- 2) Give valid credentials and login to the vtiger application
- 3) Click on Organizations tab
- 4) click on '+' (create organization) image
- 5) Give organization name and select industry from dropdown
- 6) click on save
- 7) click on organizations link
- 8) Validate if new organizations is added to organizations list
- 9) signout of the vtiger
- 10) close the browser

II CreateContactWithExistingOrganizationTest

- 1) Open the browser and type localhost:8888
- 2) Login to vtiger with valid credentials
- 3) Click on contacts tab
- 4) click on '+' (create new contact) button
- 5) Select salutation from first name dropdown
- 6) Enter lastname and select existing organization by clicking '+' button
- 7) select contact image from local disk and click on 'save' button
- 8) Click contacts link and verify if newly created contact is added
- 9) logout and close the browser

Q11) Create And Duplicate Lead Test

- 1) Open the browser and type localhost: 8888
- 2) Enter valid credentials & login to vbiger
- 3) Click on leads tab
- 4) click on '+' button (create new lead)
- 5) Select salutation from first name dropdown
- 6) Enter mandatory fields if any
- 7) click on save button
- 8) Click on 'Duplicate' button
- 9) Modify lastname and click on 'save' button
- 10) Click on leads link and verify if modified lead is added or not
- 11) logout and close the browser.

Data Driven Testing (Parameterization)

The process of fetching data from external resources like properties file, excel, database and using it in test scripts is called Data Driven Testing.

Types of Data:

1) Common Data: Data which is common to all the test scripts-

Ex: url, credentials, configuration data

This data is stored in properties file

2) Test data: Data which is test script specific

Ex: createOrganization, → lastname, industry

This data is stored in Excel

Why Data Driven Testing?

As per the rule of automation, we should not hardcode the data in test scripts. Also modifying data in test scripts might become a tedious job. Instead we store data in external resources and access it whenever required.

Advantages of DDT:

- 1) Maintenance of data is easy.
- 2) Modification of data is easy.
- 3) Run same test scripts on multiple data
- 4) Avoids hardcoding
- 5) Data sharing is easy

JDBC (Java DataBase Connectivity)

Port Number = 3306

Username : root

Password : root

Url : jdbc:mysql://localhost:3306/

MySQL commands

- 1) show databases;
displays the list of databases available
- 2) create database databaseName;
creates a new database with given databaseName

3> use databasename;

changes to given databasename

4> create table tableName (col1Name datatype(size) constraint,
col2Name datatype(size) constraint,

creates a new

table in the database

);

5> insert into tableName (col1Name, col2Name, ...) values (v1, v2, ...);

insert the values into the table

6> show tables;

displays existing tables

7> select * from tableName;

displays all columns and data in the table

8> desc tableName;

displays table description

9> delete from tableName where colName = value;

deletes particular record in the table

JDBC — Java DataBase Connectivity

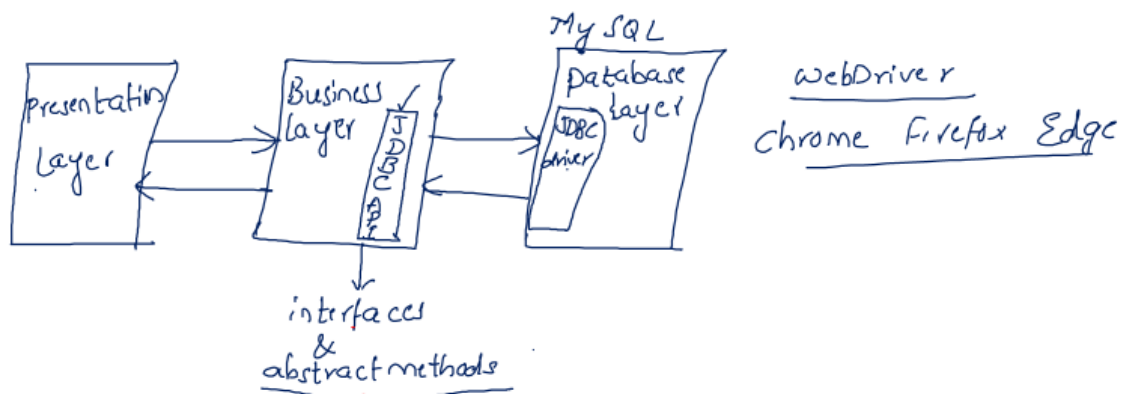
JDBC is a J2EE (Java 2 Enterprise Edition) API which is used to connect to any database using java code.

JDBC API: It is available in java.sql package, it is used to communicate to database.

- It is database independent
- It consists of interfaces and abstract methods

JDBC driver: It is the implementation of JDBC API.

- It is database dependent
- It is provided by database vendor



API — Application Programming Interface
↓
100% abstraction

Why JDBC?

Used both by developers & automation engineers.

Developers — To validate data with respect to the source code

Automation Engineers — To set pre-conditions for testing an application

Types of Databases:

Two types of databases

1) SQL Database: We store data in the form of tables.

Ex: MySQL, MS SQL, Oracle, SQL*

2) NoSQL Database: We store data in JSON format

JSON — Java Script Object Notation

Ex: NoSQL, MongoDB, Cassandra

MySQL

port No.: 3306

Username: root

Password: root

url: jdbc:mysql://localhost:3306/wcs23

To perform actions on database:

1) MySQL should be downloaded & installed

2) Get jdbc url

3) Add mysql connector/J dependency to pom.xml

Types of Queries

1) Select query — The query which has select clause

Ex: DQL statements

2) Non-select query — The query which does not have select clause

Ex: DDL, DML statements.

Properties file:

It is used to store common data.

→ The data is stored in key-value pairs

Why we store common data in properties? ✓

Common data is used in all the test scripts.

And java provides its own library to fetch data from properties file. Hence it is easier and faster to fetch data from properties file.

~~Adv~~

Advantages of properties file

- 1) It is light-weight
- 2) Faster compared to any other files
- 3) Need not add any libraries/external resources
- 4) No need of any 3rd party tools.

Disadvantages

- 1) Not organized way to store data
- 2) Can't fetch multiple data
- 3) Since everything is stored in key-value pair, searching for particular data becomes tedious job

Excel file:

Excel file is used to store test script specific data.

→ Data will be stored in the form of table

→ We use Apache POI to fetch data from Excel

→ Apache POI is a 3rd party tool which is used to fetch data from all the Microsoft documents.

Pre-requisite: Add apache poi dependency to pom.xml ✓
Add Excel workbook to project ✓

Advantages of Excel file

- 1) We can store data in an organized manner
- 2) Searching for particular data is easier

Disadvantage

- 1) It is little slow compared to properties file
- 2) We need to integrate with 3rd party tools

Generic Libraries: It is one of the framework components which is common to all the test scripts. It contains all reusable methods and classes
Generic Libraries should be created as a separate package in src/main/java

Generic libraries:

1. **IConstantPath interface** : It is used to specify file paths
 - Properties file path
 - Excel file Path
 - Photo path
 - Database Url path
2. **PropertyFileUtility class** : It is used to specify actions on Properties file
 - propertyFileInitialization()
 - getDataFromPropertyFile()
3. **ExcelFileUtility class** : It is used to specify actions on Excel file
 - excelFileInitialization()
 - getDataFromExcel()
 - getMultipleDataFromExcel()
 - getMultipleDataBasedOnKey()
 - writeToExcel()
 - closeExcel()
4. **DataBaseUtility class** - It is used to specify actions on database
 - databaseInitialization()
 - getDataFromDatabase()
 - modifyDatabase()
 - closeDatabase()

5. **WebDriverUtility class** : It is used to store all webdriver actions
 - launchbrowserAndMaximize()
 - navigateToApplication
 - implicitlyWait()
 - explicitlyWait()
 - Mouse Actions methods
 - Dropdowns
 - Scrollbar
 - popups
 - Keyboard
 - frames and windows
 - closebrowser()

Advantages of Generic Libraries:

1. Reusability of code
2. Testscript optimization
3. Avoid duplicate codes
4. Code readability increases
5. Need not remember syntax, create once and utilise it number of times
6. Saves time, test script development will be faster

Data Driven Testing Using XML:

XML - Extensible Markup Language

XML is used to enable communication and data transfer between two applications.
It is advanced level of HTML.

Debugging: To understand the root cause of error we go for debugging.

What is debug mode?

To run test script in slow mode based on user instruction is called debug mode.

■ -> Break point : It pauses the execution in break point at the time of execution

f5 -> Transfers control from calling method to called method

f6 -> Executes current line and transfers control to next line

f7 -> Transfers control from called method to calling method

ctrl + f2 -> Terminate the execution

f8 -> Resume the execution from the break point to another break point in faster mode

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

1. **implicitlyWait** - driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

2. **explicitlyWait** - WebDriverWait w = new WebDriverWait(driver,Duration.ofSeconds(10));
w.until(ExpectedConditions.visibilityOf(element));

3. **FluentWait** - It is Selenium wait which makes it possible to modify polling period

It also defines how frequently WebDriver should check if the condition appears before throwing ElementNotVisibleException

syntax:

```
FluentWait w = new FluentWait(driver)
    .withTimeOut(timeout,SECONDS)
    .pollingEvery(timeout,SECONDS)
    .ignoring(Exception);
```

4. Custom Wait: Custom Wait is user defined wait statement.

JavascriptExecutor:

JavascriptExecutor is an interface having 2 abstract methods.

1. executeScript()
2. executeAsyncScript()

Different actions that can be performed using JavascriptExecutor:

1. Scroll - hardcoding the coordinates
 - Using Location
 - Using element reference
2. Open an application
3. Pass data to the element
4. Click on the element
5. Refresh the page
6. Get title and Url of the page
7. Handle disabled element

ChromeOptions: It is a class in Selenium which is used for customizing chrome driver sessions.

We can perform following operations using ChromeOptions:

1. Disable Notifications
2. Run browser in maximized mode
3. Headless mode - browser without UI, execution in headless mode is faster
4. Incognito mode

POM(Page Object Model):

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

Why Object Repository?

In Agile methodology due to frequent requirement changes GUI changes frequently. Hence we cannot hardcode the element references. Also everytime changing element references in the test scripts can become tedious job. Therefore we use Object repository to store all the elements.

POM: It is one of java design patterns preferred by Google to develop Object Repository. POM has 3 stages:

1. Declaration: `@FindBy(LN="LV") private WebElement element;`
`@FindBy(LN="LV") private List<WebElement> element;`
2. Initialization: We call parameterized constructor.

```
public Classname(WebDriver driver){
    PageFactory.initElements(driver,this);
}
```
3. Utilization: We provide business libraries to all the elements declared.

Advantages of POM:

1. It handles StaleElementReferenceException
2. Reusability of elements and respective methods
3. No need to write xpath again and again
4. Modification of elements is easy
5. Maintenance of elements is easy
6. Test script optimisation is achieved
7. Development of test script is faster
8. POM supports Auto-Healing feature

@FindBy:

- It is Selenium annotation

Syntax -

```
@FindBy({@FindBy(LN1="LV1"),@FindBy(LN2="LV2"),...,@FindBy(LNn="LVn")})
private WebElement element;
```

- It works like AND operator
- Here we give multiple locators to find element, it returns
WebElement/List<WebElement> only if all the @FindBy() return the WebElement.

@FindAll:

- It is Selenium annotation

Syntax -

```
@FindAll({@FindBy(LN1="LV1"),@FindBy(LN2="LV2"),...,@FindBy(LNn="LVn")})
private WebElement element;
```

- It works like OR operator.
- Here it tries to locate the element using @FindBy one by one. If the element is located then it returns element address neglecting remaining @FindBy

- It supports Auto-Healing feature.

Auto-Healing: In automation, if one locator fails to locate an element then it automatically checks with the next locator to find the element. This process is called Auto-Healing.

TestNG (Test Next Generation): TestNG is a unit testing framework tool used by both developers and automation engineers.

We have other testing tools:

Java - JUnit
.net - NUnit
Python - PyDev
Javascript - Jasmine/ mocha

TestNG is the combination of JUnit and NUnit.
It supports TDD (Test Driven Development) framework.
TestNG is developed as a plugin to the IDE.
TestNG does not have UI. IDE provides UI for TestNG

Developers Usage: Developers use TestNG for unit testing or White box testing
Automation Engineers Usage: To develop a framework and to do batch execution

Advantages of TestNG:

1. Download and install TestNG easily since it is an open source tool
2. We can set priority to the test cases
3. Invocation count
4. enabled = false
5. dependsOn method
6. Annotations
7. @DataProvider
8. @Listeners
9. Batch execution
10. Group execution
11. Parallel execution
12. Generate reports
13. Re-run failed test cases
14. Assertions

Annotations: It is a Java template which starts with '@'. It is used to provide meta data to JVM. These annotation methods need not be called.

@BeforeSuite: It executes before <suite> tag in xml file. Generally we provide configuration steps before running a test suite.

Ex: Database configuration

@BeforeTest: It executes before <test> tag in xml file. It is used for parallel executions. It generates multiple threads and executes test scripts in parallel.

@BeforeClass: It executes before <class> tag in xml file. It is used to initialize utility files and launching the browser.

@BeforeMethod: It executes before @Test method. Here we generally initialize all POM classes and login to application is done.

@Test: It acts like a main method in Java. All the executions start here. Whenever JVM encounters @Test it checks if there are any @Before methods and @After methods and carries the execution accordingly.

A class can contain more than one @Test methods.

Generally a class can have 15 @Test methods

@AfterMethod: It executes after the execution of @Test method. Generally here we give logging off action.

@AfterClass: It executes after </class> tag in xml file. Here we give closing the browser and excel.

@AfterTest: It executes after </test> tag in xml file.

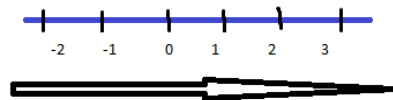
@AfterSuite: It executes after </suite> tag in xml file.

Priority: It is used to set the priority to the test cases. We should pass 'priority = int_value' in @Test. Default priority is 0.

If test cases have same priority, they run according to the ASCII values of their names.

Usage: @Test(priority=1)

Priority follows integer number line order



Invocation count: In order to execute test script multiple times with same test data we use Invocation count. It acts like a looping statement.

- Default invocationCount is 1.

- If the invocation count is 0 or negative, it will not consider the execution of test script

Enabled=false: It disables the test script execution. By default enabled is true.

dependsOnMethods: In order to make a test script depend on the other test script we use dependsOnMethod.

Usage: @Test(dependsOnMethods = "testName")

@Test(dependsOnMethods = {"test1", "test2", ...})

Batch Execution: Executing all the test scripts sequentially/executing test suite is called batch execution.

Step 1: Select all the test scripts -> Right click -> TestNG -> Convert to TestNG

Now all the test scripts are converted to xml file.

Step 2: Run the xml file

Generally we run full regression test suites in batch.

Group Execution: Grouping the similar test scripts and executing is called group execution.

To achieve the group execution:

1. We should specify the group for each test script and each annotation

@Test(groups = "Smoke")

@Test(groups = {"Smoke", "Sanity"})

2. Convert the test scripts to xml file

3. Give <groups> tag after <suite> tag in xml and specify which group of test cases should be executed within the groups tag

If the group is not specified in annotation it will not consider the block for execution

Generally used for Regional Regression testing

alwaysRun = true: This is used to run the block irrespective of groups

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

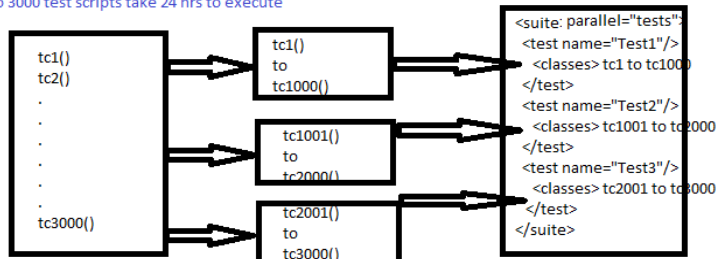
```
<suite>
  <test>
    <classes>
      <class>
        <methods>
          <include>
          <exclude>
        </methods>
      </class>
    </classes>
  </test>
</suite>
```

Parallel Execution: We have 3 types here

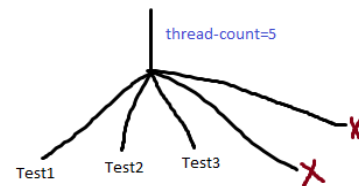
1. Distributed parallel execution
2. Cross Browser parallel execution
3. Cross platform parallel execution

1. Distributed parallel execution:

Let's assume there are 3000 test scripts and 1000 test scripts takes 8 hrs to execute.
So 3000 test scripts take 24 hrs to execute



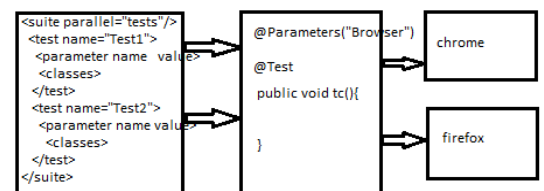
- > Here we distribute test scripts to different <test> tags
- > We enable parallel="tests"
- > Each <test> should be given unique name



- > thread-count should be equal to number of <test> tags
- > By default thread-count is 5

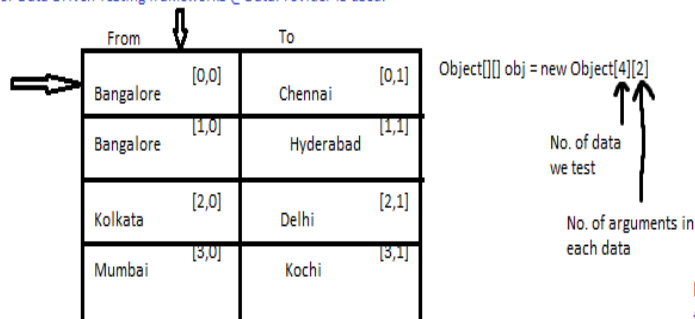
Cross Browser Parallel execution/ Browser Compatibility testing:

- > We execute same set of test scripts on different browsers.
- > To achieve cross browser parallel testing
 1. We should enable parallel="tests"
 2. Within <test> we should give <parameter> tag specifying the browser
 3. Before all the annotations we should provide @parameters() in test script



@DataProvider: To run same test script multiple times with different test data we go for @DataProvider.

- > Data should be stored in two dimensional Object array
- > Return type is two dimensional Object array
- > Each test script should have dedicated @DataProvider
- > It is used in the applications where huge data is to be tested on the application like banking, ecommerce, ticket booking.
- > For Data Driven Testing frameworks @DataProvider is used.



Assertions: Assertions are used to validate the test script.

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

Normal if-else statements doesnot have the capability to fail the test scripts since depending upon the condition specified one of the if-else block gets executed.

In Assertions, If assertion statement fails, it throws AssertionError providing the line number where actually the error occurred.

We have two types of Assertions:

1. Hard Assert/Assert: Assert is a class and all the methods in Assert are static.

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

If assertion statement fails it stops executing the current block where the error has occurred and transfers the control to next block and also throws AssertionError.

We use Hard Assert for mandatory fields.

Soft Assert: Soft Assert is a class and all the methods in Soft Assert are non-static.

Different methods in SoftAssert are:

```
SoftAssert s = new SoftAssert();
s.assertEquals()
s.assertNotEquals()
s.assertTrue()
s.assertFalse()
s.assertNull()
s.assertNotNull()
s.assertAll()
```

If assertion statement fails, then it will not stop executing current block. It will continue the execution of remaining statements in the current block and then transfers the control to next block.

assertAll() method is mandatory and it should be given at the end of each block.

We use SoftAssert for the fields where major or minor defects occur.

Listeners: It is a feature in TestNG which monitors the actual test execution and captures the real time events during the execution and performs necessary actions depending upon the success or failure or skipped status of the test scripts.

Listeners have different interfaces:

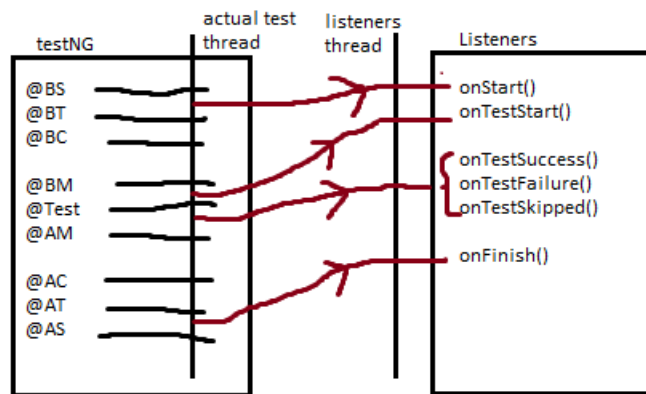
1. ITestListener
2. ITestResult
3. IRetryAnalyzer
4. IAnnotationTransformer

1. ITestListener: It enables us to capture pass or fail of test script.

To achieve this we have to provide implementation for all the methods present in ITestListener interface and we should provide @Listeners before each @Test

Methods in ITestListener interface are:

onStart(), onStartTest(), onTestSuccess(), onTestFailure(), onTestSkipped(), onFinish()



Listeners implementation class should be given in `src/main/java` in `genericLibraries` package.

ITestResult: It is used to record the status(pass/fail) of the `@Test` method. It is used as argument to the listener methods.

IRetryAnalyzer: It is used to re-run the failed test scripts.

Sometimes test script might fail due to synchronization issues or server issues. When the failure occurs we re run the failed test scripts manually. To avoid this manual work we go for `IRetryAnalyzer` implementation.

Step 1: Provide implementation to `IRetryAnalyzer`

Step 2: Provide the qualified path of the `IRetryAnalyzer` implementation in `@Test`

`IRetryAnalyzer` implementation class should be given in `src/main/java` in `genericLibraries`

IAnnotationTransformer: It will transform one annotation type to another annotation.

Step 1: Provide implementation to `IAnnotationTransformer` interface

Step 2: Convert the test script to testNG xml file and in xml provide qualified name of `IAnnotationTransformer` implementation class

BaseClass: `BaseClass` contains all before and after annotations.

It should be given in generic libraries in `src/main/java`.

All the test scripts should extend to `BaseClass`

Reports: TestNG generates html reports. Reports act as a proof that we have executed all the test scripts.

Two types of reports:

1. High level reports: TestNG generates html reports. These are high level reports.

It contains only

-> Test script name

-> Status - pass/fail/skipped

-> Exception name and the line number where it has occurred

2. Low level reports: The log level analysis is called low level reports. In TestNG we have Reporter.log()

Disadvantages of html reporting in testNG:

1. We cannot attach screenshots of failed test cases.

2. We cannot customize report structure

3. We cannot have graphical representation of the reports

Extent Reports: Extent reports are advanced reporting widely used in many organizations.

It gives graphical representation of high level reports. We can customize report structure.

It has mainly 3 classes.

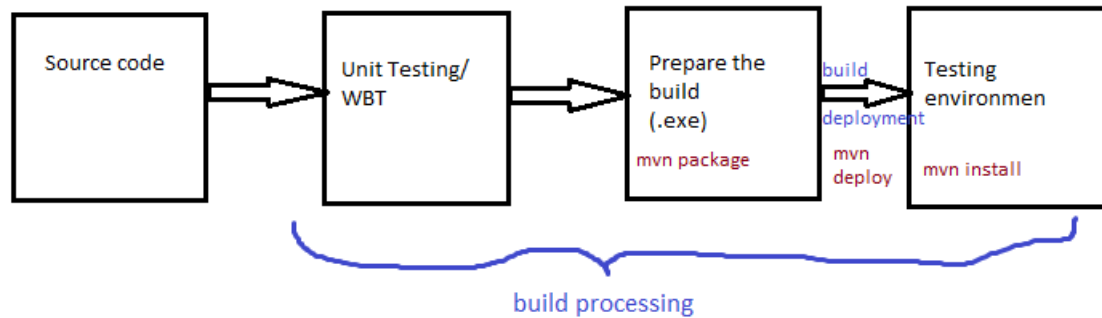
ExtentSparkReporter: Used to set basic configuration of report.

ExtentReport: Used to set system info in the reports

ExtentTest: Used to set test level analysis

ExtentReports can be configured either in BaseClass or Listener implementation. But Listener implementation is preferred since it monitors test execution.

Maven: Maven is build management tool/ build process tool/build deployment tool.



Why maven in development?

In development maven is used for build processing like build testing, build creation and build deployment

Build testing: Testing the compilation issues in project and providing the status

Build creation: Converting source code to .exe file

Build deployment: installing the build into other environment

Why maven in automation?

In automation framework we use maven to clean, validate, compile and test all the test scripts in the framework

Advantages of maven:

1. Checks the integration issues of framework components
2. Handles dependencies jars
3. We can run test scripts in cmd line
4. It allows profiling
5. It supports Jenkins
6. Creates quick configuration set up
7. It supports cmd line parameters

Softwares required for Maven project:

Maven Eclipse plugin: It is inbuilt plugin in eclipse which allows us to create maven project and provides us with the folder structure.

1. src/main/java: Generic libraries, object repository
2. src/main/resources: driver executable files, Framework User Guide, CRS
3. src/test/java: Test scripts
4. src/test/resources: commonData.properties, TestData.xlsx
5. pom.xml: (Project Object Model) It is the heart of the framework since it contains all the dependencies and plugins.

Dependencies: It is the advanced feature in maven which automatically downloads all the jar files needed for the framework depending upon the dependency added.

Maven Surefire plugin: It should be added explicitly by the user into the framework in pom.xml. It is used to execute the test scripts whose names are appended with "test".

Maven Build Life Cycle: While in real time, many automation engg work on the same framework. Some times if the changes made by one automation engg might effect the entire framework. To avoid this situation we run maven build life cycle.

mvn clean: It is used to clean all the older reports in target folder

mvn validate: It is used to validate the framework. It checks if all the jar files and plugins are present in the project or not. If not it downloads the files automatically.

mvn compile: It is used to check the compilation issues in the framework.

mvn test: It is used to execute the test scripts that has "test" appended to their class names and method names.

Other maven commands:

mvn package: It is used to create the build

mvn install: It is used to install the build in other environment

mvn deploy: It is used to deploy the build to another environment

To run specific class in cmd line:

`mvn test -Dtest=classname`

To run multiple classes in the test suite:

`mvn test -Dtest=classname1,classname2`

To run specific methods in the class:

`mvn test -Dtest=classname#methodname`

Parameterization in maven cmd line:

We can pass parameters in cmd line for test execution.

Sometimes customer might want to run the tests in different browsers or with different credentials. So maven provides us with this feature to pass parameters in the cmd line during run time.

`mvn test -Dkey1=value1 -Dkey2=value2`

In the script we should get this parameters using `System.getProperty(String key);`

Right click on test script -> Run -> Run configurations -> Arguments -> VM arguments -> pass parameters (-Dbrowser=chrome -Durl="http://github.com")

Maven profiling: Executing specific test suite files depending on the profile mentioned in the maven command line is called maven profiling.

In order to achieve this we should create profiles with different id in pom.xml file. Since by changing the xml file name in <suiteXmlFile> tag in Surefire plugin several times might corrupt the pom.xml file. If pom.xml file is corrupted then there is no other way than discarding it.

To run specific xml test suite file:

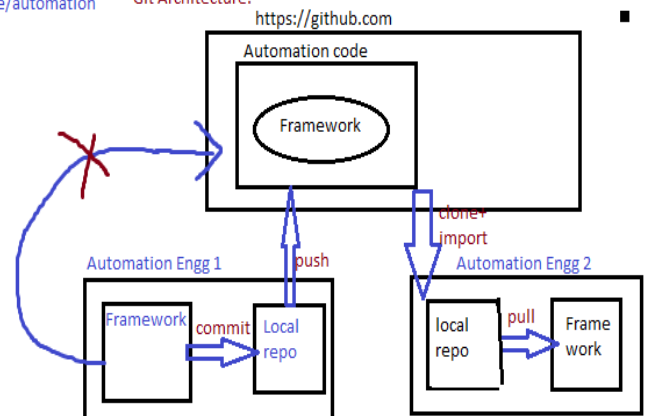
mvn test -P moduleName

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

GitHub: GitHub is Version Control tool, Code Management Tool, Build Management Tool.

It is a distributed decentralized cloud based repository which is used to maintain the source code/automation framework/CRS document/build versions in one place.

Git Architecture:



Advantages:

1. Since it is cloud based repository maintenance team is not required.
2. Since it is cloud based repository we can access the repository via internet from any place.
3. Cloud means pay for what you use.
4. Cloud means contributors can access the code from remote places
5. It provides a platform for File sharing among the team members
6. It provides way to solve conflicts in code
7. Jenkins also gets the new build from github
8. Scale up/scale down is easy.
9. It also provides history of changes made in the framework
10. No initial investment needed for storage
11. It provides a way to review the automation framework.

Softwares in GitHub:

1. **GitHub:** It is a cloud based repository which can be accessed via <https://github.com>
We should create an account in github.com
2. **GitClient:** It should be installed in our local system. This takes care of the local repository.
Ex: **Egit:** It is available in eclipse.
- GitBash:** It should be installed in local disk. This is used to perform all operations via commandline

Developer Usage: Developers use github to maintain their source code

Automation Engg Usage: To maintain automation framework

Devops Usage: To maintain build version files

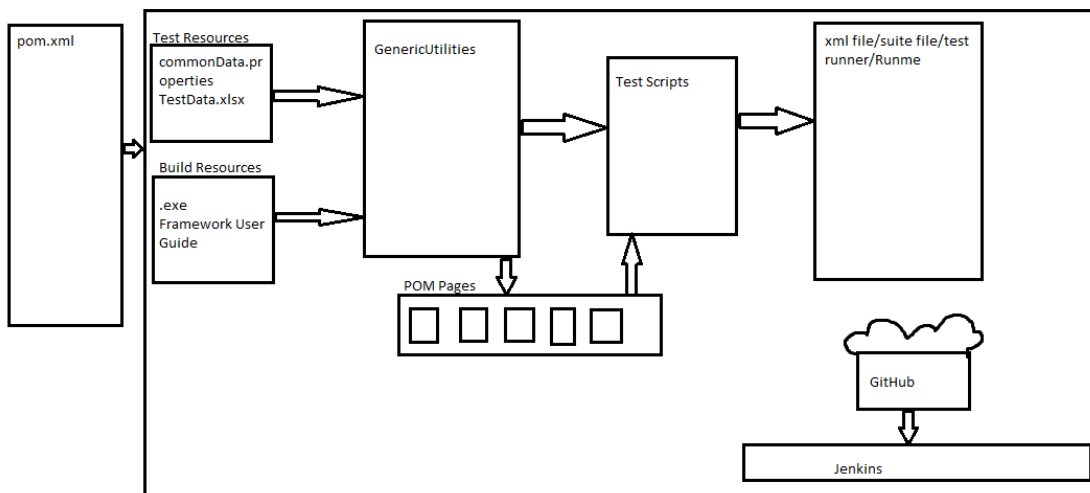
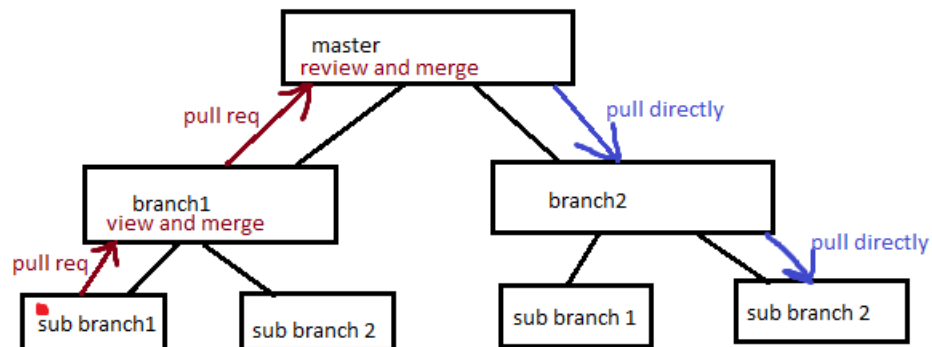
Manual testers: To maintain CRS document

Why github is a decentralized repository?

When the code is to be pushed to global repository it initially creates a local repository and stores the code first and verify if the code is working fine or not. Only then it is pushed to global repository.

Branching in GitHub: Since many automation engineers work on the same Automation framework, chances are there that the framework might get corrupted. Hence we go for branching.

Here we can create branches from the master branch and also sub branches from the branches.



Frameworks: It is the collection of reusable components which makes test script development faster and also test execution is easier and also provides with generation of reports automatically.
It is well organized structure.

Different framework approaches:

1. TDD(Test Driven Development):

- Test cases are mandatory
- @Test is the main driving factor
- TestNG in TDD frameworks
- We have many reusable methods

2. BDD(Behaviour Driven Development)

- Test Scenarios are mandatory
- Automation test engineer will develop the test cases using feature file
- Step definitions are given to the steps in feature file
- Runner file is the place where actual execution takes place
- There won't be many reusable methods
- We use BDD Cucumber tool

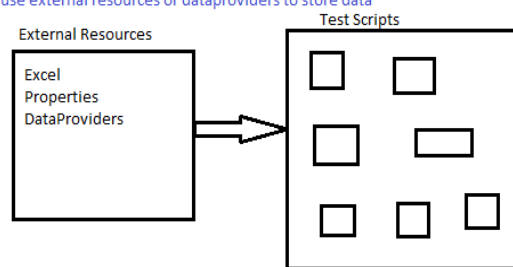
Types of frameworks:

1. Data Driven framework: When the data to be tested is huge compared to the methods then we go for Data Driven Framework.

Ex: Ecommerce, Banking

- Here data is the driving factor

- We use external resources or dataproviders to store data



4. Keyword Driven Framework: The automation engineer/ developer will be providing keys in Excel files or CSV files. Also they develop the program for the keys. These keys are used by test engineers to automate the test scripts. This type of framework is called as Keyword Driven Framework.

Even the manual testers can automate the test scripts.

It is used only for small applications. Ex: Education

Test Data		Object Repo	
action	input	element	locator
type	admin	UN	id='user'
click	--	pwd	name='pass'
drag	items	submit	type=submit

5. Hybrid framework: It is the combination of two or more frameworks.

Data Driven + Modular Driven

Data Driven + Method Driven

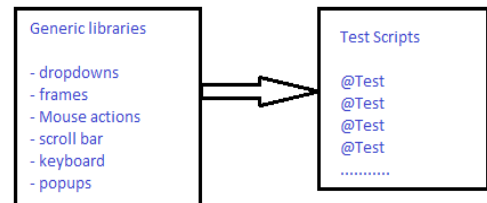
Data Driven + Keyword Driven

Data Driven + Keyword Driven + Modular Driven

Data Driven + Method Driven + Modular Driven

Method Driven + Modular Driven

2. Method Driven Framework: When the application has lot of reusable methods we go for method driven framework.

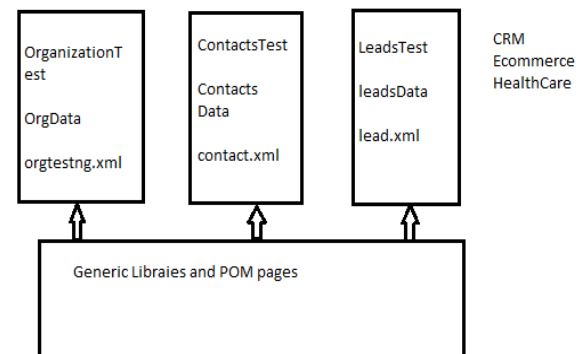


- methods are driving factor

Ex: CRM, ERP

3. Modular Driven Framework:

When we have huge number of modules then we store test cases, test data and xml files module wise. This is called modular driven framework.



Stages in Framework :

- 1. Choice of framework:** Technical leads, Architects (10-15 years exp), Managers will decide which framework to be adopted to the application
- 2. Framework Design:** It includes development of Generic libraries, test data templates, POM classes
- 3. Framework Development:** POM classes implementation and test script development by test engineers will be done in this stage
- 4. Framework execution:** The framework will be pushed to GitHub and test execution is carried on in Jenkins by technical leads.

Framework Advantages:

1. It provides organized folder structure
2. It easy to handle dependencies and plugins
3. Easy to maintain and modify test data
4. Easy to maintain and modify element repository
5. It provides reusable methods
6. Debugging is easy
7. Screenshots of failed test scripts can be taken
8. Generates accurate reports
9. It supports batch, group, parallel executions
10. Test Script optimization
11. Test script development is faster

Selenium Grid:

It is the combination of Selenium WebDriver + Selenium RC+ Selenium Grid Server.
It is the open source tool which facilitates remote executions and also compatibility testing.

Remote Execution:

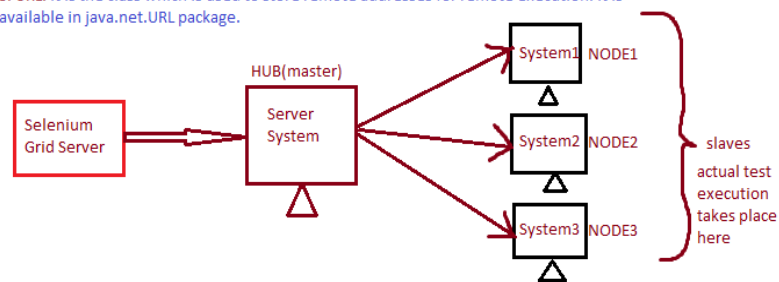
Executing test scripts in remote devices like computers, cloud computing is called remote execution

Pre-requisites:

1. JDK
2. IDE
3. Selenium Server
4. Selenium standalone server
5. Driver executable files

How to perform remote execution?

1. **RemoteWebDriver:** It is the concrete implementing class of WebDriver interface which is used to perform all remote execution using Selenium Grid. It is available in `org.openqa.selenium.remote.RemoteWebDriver` package.
2. **DesiredCapabilities:** It is the class which is used to set or change the capabilities of webdriver. During execution we need to set the browser name, platform and OS etc. It is available in `org.openqa.selenium.remote.DesiredCapabilities` package.
3. **URL:** It is the class which is used to store remote addresses for remote execution. It is available in `java.net.URL` package.



Selenium Grid : Used to perform - Remote Execution, Cross Browser Execution, Cross Platform Execution

HUB: It is server system which acts like master and bypasses the command from Selenium server to the nodes.

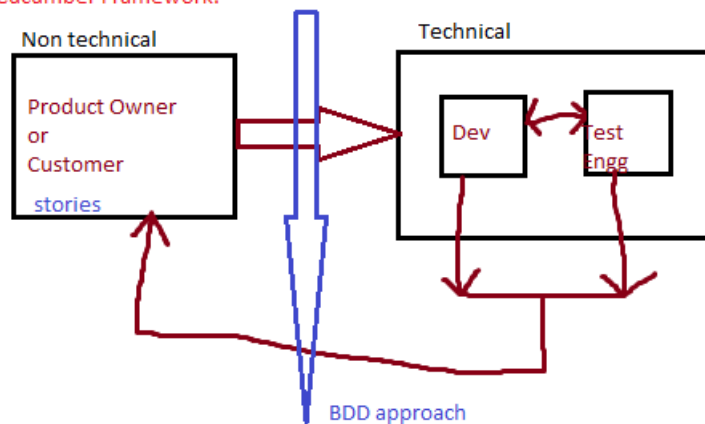
NODE: Nodes are the place where actual test execution takes place. They take the command from HUB and execute the test script.

How to create HUB and NODE and perform execution?

1. Open cmd prompt and type the following command to register hub node
`java -jar <seleniumstandaloneserverpath> -port <portNumber> -role hub`
2. Open another cmd prompt and type the following command to register to node
`java -Dwebdriver.chrome.driver=<pathofchromedriverexecutable> -jar <seleniumstandaloneserverpath> -port <portNumber> -role node -hub <hubIpAddress>`

Default port number is 4444

BDD Cucumber Framework:



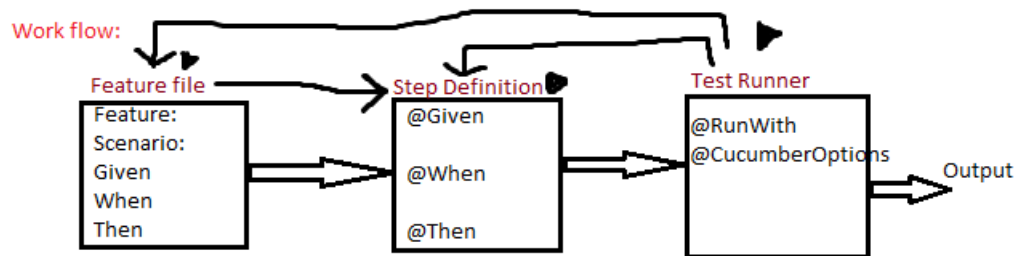
Fills the gap between non technical and technical people in an organization

Cucumber Tool:

It is an open source tool which is used to write the test cases using feature files and create step definitions for every step in feature files and execute these feature files via test runner class and generate the reports.

Steps to develop cucumber framework:

1. Add Cucumber plugin to eclipse
 - Go to Help -> Eclipse Marketplace
 - Search for cucumber and install the plugin
2. Create Maven project
3. Add dependencies
 - cucumber java
 - cucumber junit
 - junit
 - selenium java
 - webdrivermanager
4. Create a folder for feature files in `src/test/resources`
5. Create new file with .feature extension in feature files folder



Gherkin language: It is simple English language which has predefined keywords which helps to write the test scenarios in organized manner.

Few Gherkin Keywords:

1. **Given:** It is used to give the precondition before test is started
2. **When:** It is used to give the condition in the test, actual test steps are given here.
3. **Then:** It is used to give validation for test case
4. **And:** It is connecting keyword which connects two or more conditions or validations or pre conditions

Note: Gherkin keywords always start with Uppercase letters.

Parameterization In Cucumber:

For parameterization we use Scenario Outline instead of Scenario.

In Scenario Outline we pass parameters as 'Examples' along with the scenario steps.

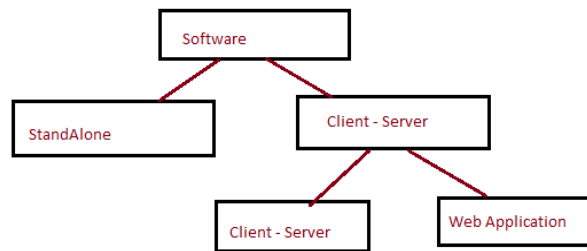
Reports in Cucumber Tool:

1. **HTML Reports:**
`plugin = {"html:target/HTMLReports/report.html"}`
2. **JSON Reports:**
`plugin={"json:target/JSONReports/report.json"}`
3. **JUNIT Reports:**
`plugin={"junit:target/JUNITReports/report.xml"}`

The above should be added to CucumberOptions in test runner classes to get the reports.

Introduction to software:

An application/ software is set of instructions which is designed to perform specific task.



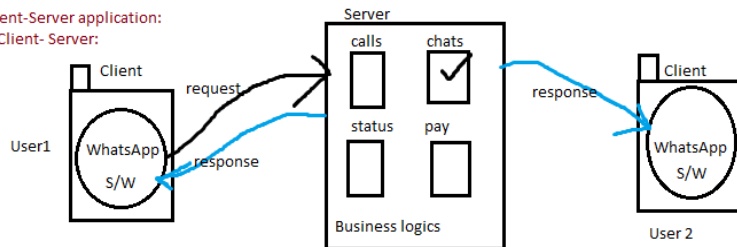
StandAlone application:

An application which can be downloaded and installed and can be accessed without any internet or browser is called stand alone application

Ex: Calculator, Paint, Calender etc

Client-Server application:

1. Client- Server:



The software is downloaded and installed in the local device. This software acts like client. When user accesses this software, request will be sent to the server. Server processes the request and sends response.

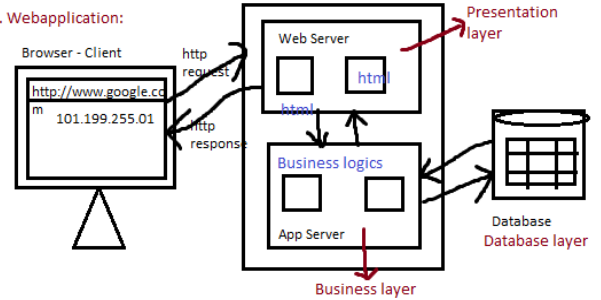
This type of application is called as client-server application.

Here we access application via internet. But browser is not required.

Ex: WhatsApp, Instagram, Facebook, Flipkart, Zomato, Ola etc

Server 101.199.255.01

2. Webapplication:



An application which is accessed via internet through browser is called web application.

Multiple users can access the application at a time.

Web application can be accessed only via browser.

Components of Webapplication:

1. Web Server
2. App Server
3. Web Browser
4. Database

1. **Web Server:** Like any other application web server is also an application which runs on Operating System and every web application is under control of web server

Web server serves the request to a web application. It helps both web browser and web application to interact with each other.

Ex: Apache TOMCAT, Apache JBOSS, Oracle Weblogic, Oracle Glassfish

2. **App Server:** It contains web resources. Web resources are the business logics of source code. Hence app server is also called as Business layer. To implement these resources servlets or JSP for java, Asp.net for C# are used.

3. **Web Browser:** It is stand alone application which is used to access web application.

Browser understands only html so it always creates http request and interprets http response received from web server.

It is one and only one way to access web application.

Ex: chrome, opera, safari, edge, firefox etc

4. **Database:** It is also an application which is used to store data.

Ex: MySQL, NoSQL, MsSQL, Oracle 10g, MongoDB etc

Web URL:

URL - Uniformed Resource Location

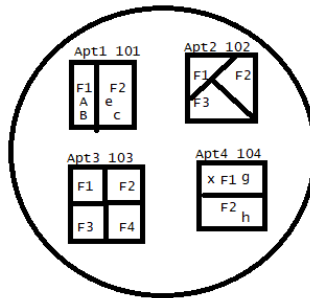
- It is the one and only one way to uniquely identify the web resource in a web application.
- It has maximum 200 characters

Syntax:

protocol://domainName:portNum/resourcePath?QueryString#fragmentID

address://Apt4:F1/g
address://104:F1/g

address://Apt1:F2/c
or
address://101:F2/c



Protocol:

- It is common language where two applications exchange information
- It is set of rules and instructions which enables the communication between two applications
- It is optional information and it is case insensitive.

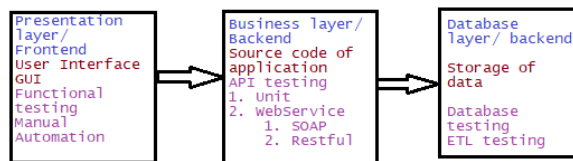
Types of Protocols:

1. http - hyper text transfer protocol
2. https - hyper text transfer protocol secure
3. smtp - simple mail transfer protocol
4. ftp - file transfer protocol

Web Services:

Any service which is rendered over internet or web is called web services.

SOA(Service Oriented Architecture):



API Testing:

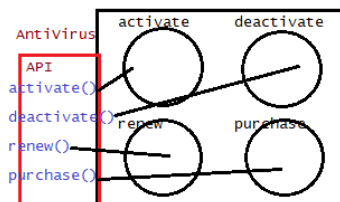
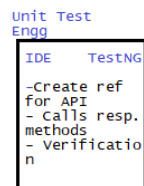
API - Application Programming Interface

- Testing an application in Business layer.
- Testing an application without browser.
- It is a type of software testing where we test all the APIs of the application and determine if they meet expectations for functionality, performance security and reliability.

Types of API Testing:

1. Unit API Testing:

- Testing the source code of the application using another program
- Testing the source code of the application using an API
- Testing the untouched part of the source code of an application using an API



Domain Name:

- It is used to identify specific computer on the network in which required web application is present.
- Domain name might be computer name/IP address.
- It is mandatory field and case sensitive.

Ex: www.facebook.com, www.amazon.com

Port Number:

- It is used to identify specific software/ application inside the computer.
- It is not mandatory field.

Ex: http://localhost:8888
http://localhost:8080

Resource Path:

It is used to uniquely identify specific web resource in the application

It is optional information

Ex: https://www.amazon.com/gift-cards

https://www.facebook.com/login

Query String:

It is name and value pair which is used to carry information from client to server or browser to application.

It is optional information

There can be more than one query string in separated by '&'.

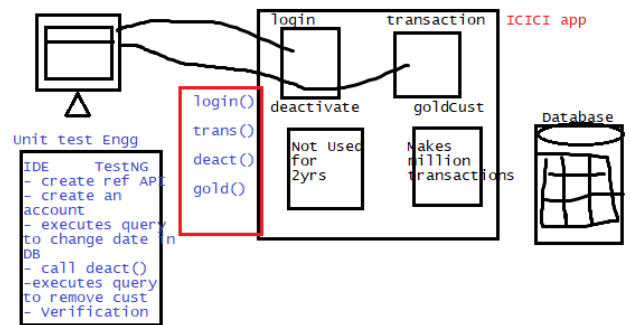
Query string always starts with '?'.

Ex: https://www.google.com/search?q=iphone

Fragment ID:

It is used to identify specific fragment in web page.

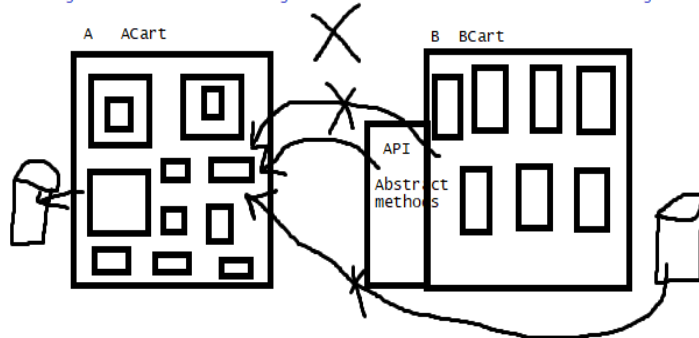
It is optional information.



2. Webservice API testing:

webservice is a mechanism/ media to enable data exchange between two applications irrespective of platforms.

Testing the web service through APIs is called webservice API Testing.



Why Webservice API Testing?

The purpose of Webservice API testing is to verify that all the APIs that are exposed by your application are working as expected or not.

Testing the request and response of the API is called web service API testing.

Webservice provider should perform API testing to ensure that all the APIs are working functionally well and performance wise well.

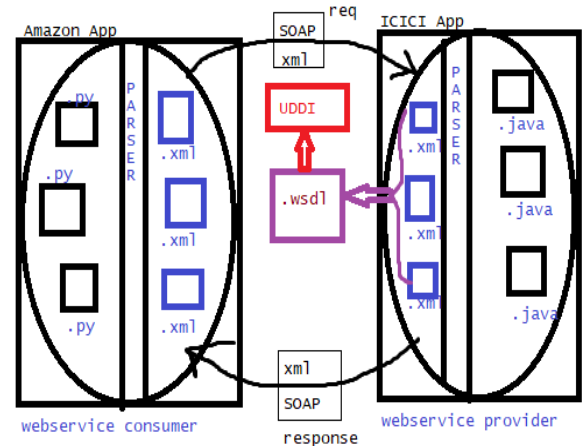
1. SOAP webservice API testing:

SOAP: Simple Object Access Protocol

It is simple xml based protocol which is used to store and exchange information between two applications over http.

SOAP is platform independent.

SOAP is language independent.



WSDL: Web Service Description Language

WSDL document describes a SOAP Webservice. It specifies the location, methods of the web service. It contains following elements:

1. type: XML Schema/ datatype used
2. message: defines data elements for each operation
3. port type: describes the operations that can be performed
4. binding: Defines protocol and data format for each port type

UDDI: Universal Description, Delivery Integration

It is xml based standard for describing, publishing and finding web service.

Serialization: The process of converting any programming language to xml is called serialization/ marshalling.

Ex: JAXB.jar, JAXWS are the parsers used to perform serialization

Deserialization: The process of converting xml to any programming language is called deserialization/ unmarshalling.

2. Rest webservice API Testing:

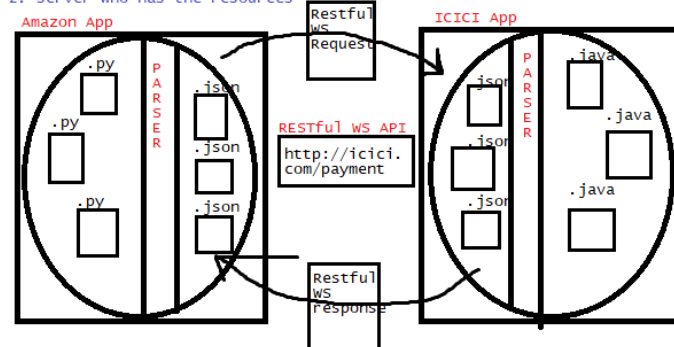
REST - Representational State Transfer

When Restful API is used, the server provides response based on the representation of the request.

It is architectural type of developing web services.

It has only two elements:

1. client who requests for resources
2. server who has the resources



Differences between SOAP and REST

SOAP	REST
1. SOAP is a protocol	1. Rest is Architectural style
2. SOAP exposes its webservices via WSDL files in UDDI	2. Rest exposes its webservices via URI
3. SOAP permits XML data format only	3. Rest permits different data formats - text,html,xml,json, js
4. SOAP has too much standards complicated to use	4. It doesnot have too many standards, easy to use
5. SOAP requires more bandwidth and resources	5. It requires less bandwidth and resources compared to SOAP
6. SOAP defines its own security	6. Inherits security from transport protocol Bearer token, oauth 1.0, oauth 2.0

Webservice API Testing tools:

1. SOAP WS API testing tool
 - SOAP UI
 - Ready API
 - WIZDL
 - SOAP Sonar etc
2. REST WS API testing tool
 - PostMan
 - RestClient
 - http master
 - Rest Assured
 - Ready API
 - Karate
 - Katalon etc

Advantages of Webservices:

1. Application, Platform and Technology independent
2. Loosely coupled - Each application is independent of one another. Changes made in one application will not have impact on other.
3. It saves development time
4. It saves money for development
5. Helps improve business partners
6. Webservices can be reused
7. Provides security to applications

PostMan:

PostMan is REST WS API testing tool. It is started in 2012. It is developed as chrome plugin in the beginning. Later a separate GUI app is developed for PostMan.

It is GUI API testing tool where we can create a request and it provides lot of features to verify the response header and body.

It is API client tool which is used for fulfilling the requirement of developer and test engg throughout API Development Life Cycle.

Developer Usage:

1. Publish: Using Postman we can generate API documentation automatically
2. Monitor: Create automated test to monitor API periodically and check for response
3. Debug: PostMan console is designed to help debug PostMan collections
4. Mock: Mock server allow you to simulate your API data

Testing Usage:

1. Send requests easily
2. Test runners allow us to run all the collections easily
3. Modify different types of parameters
4. Provides code snippets to validate responses
5. Integrates with Jenkins
6. We can import and export collections

CRUD operations in PostMan:

http methods:

POST: To create a resource in the application
GET: To get or read all the list of resources in application
PUT: create/ update(complete update) the existing resources
PATCH: To update part of the resource
DELETE: delete a resource from the application

Request URL = Base URI + Endpoint

Status codes:

We have five layered status codes

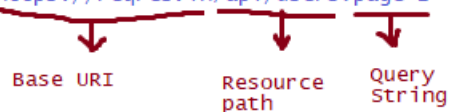
1. 100 -> continue
2. 200 -> successful requests
3. 300 -> Redirection
4. 400 -> Client side error
5. 500 -> Server side error

Variables:

1. Environment variables: Scope is environment
2. Global variables: Scope is entire workspace
3. Collection variables: Scope is entire collection
4. Local variables: Scope is within the request

Parameters: To avoid hardcoding of requests we use parameters.

https://reqres.in/api/users?page=2



1. path parameter: parameter associated with the resource path
/:{path}
2. query parameter: parameter associated with query string
3. form parameter: parameter associated with form data.
Form data is the key-value pairs in the request body.

Authentication:

Providing access to a particular API using some standards like username and password or tokens etc is called authentication.

Why Authentication?

REST donot have any security standards. It inherits security standard from application.

Moreover json is less secured compared to xml. Hence all RESTful webservices have to adhere to authentication protocol.

Authentication standards in Rest are:

1. Basic Auth
2. Bearer token
3. Oauth 2.0
4. Oauth 1.0

1. Basic Auth: It consists of Username and Password

- API developer will provide Username and password
- Minimal security as username and password are readable and can be easily shared.

2. Bearer Token:

- It is generated with help of username and password
- Token is non-readable format
- Token is given by API developer
- Token can be generated and used several times
- It is secured compared to Basic Auth but still it can be shared

Step 1: Token Generation

In Github,
profile -> Developer settings -> Personal access tokens -> Generate New Token -> Select both repos and delete repo scopes -> generate token

Once token is generated copy it to some text file

Step 2: Use the token to access Github APIs

Go to Github APIs -> Expand Overview -> Click on resources in REST API -> Copy the Base URI : <https://api.github.com>
Click on Github app enabled endpoints -> scroll until repos -> Click on second link -> Click on create repository for authenticated user -> copy the end point

POST: end point - /user/repos

request body -

```
{
  "name": "anyReponame"
}
```

OAuth 2.0:

- It is one of the authentication standard which provides one level security.
- API developer will give Client ID and Client Secret
- With the help of Client ID and Client Secret OAuth app will generate token
- OAuth app is third party app which generates access token

Steps to generate and use OAuth access token:

In GitHub,

Profile -> settings -> developer settings -> OAuth Apps -> Click on New OAuth App

Give Application Name: WCSA3

HomePage URL: <https://github.com>

Callback URL: <https://github.com>

Click on OAuth documentation and copy authorization URL

Authorization URL: <https://github.com/login/oauth/authorize>

Access Token URL: https://github.com/login/oauth/access_token

Go back to OAuth app registration page and click on Register application

Copy Client ID and generate Client Secret and copy it

Follow the same steps to access GitHub APIs

Execution of PostMan in cmd line:

Newman is the tool which is used to execute postman collection in the cmd line

To check if node js is installed -> `npm --version`

To install newman -> `npm install -g newman`

To check if new man is installed -> `newman --version`

To run the postman collection in cmd:

Export postman collections -> Click on 3 dots beside collection name ->

Export -> save it as .json extension

cmd> `newman run <path of json file>`

To include environment variables:

Export environment variables in json format

cmd> `newman run <path of collection.json> -e <path of env var .json>`

To generate html reports in newman

install basic html reporter

cmd> `npm install -g newman-reporter-html`

install advanced reporter

cmd> `npm install -g newman-reporter-htmlextra`

To run: cmd> `newman run <collection path> -e <env var path> -r html`

cmd> `newman run <collection path> -e <env var path> -r htmlextra`