Basic Selenium Contents

Chapter - 1

1) Introduction to Automation
   - what? why? Adv & Disadv
2) Automation tools
3) Introduction to Selenium
   - what, why, Adv & Disadv
4) Components of Selenium,
5) Languages supported by Selenium
6) Selenium Architecture
7) Download & Installation, setup project
8) Selenium WebDriver Architecture
9) Methods of WebDriver
10) HTML
11) Locators
12) Synchronization
13) WebElements & their methods
14) Maximize, minimize, navigation APIs

Chapter-2

Handling WebElements

1) AutoSuggestions
2) Mousehover
3) RightClick
4) Double click
5) Drag and Drop
6) Frames
7) Windows
8) Scroll Bar
9) Screenshot
10) Popups
    - Alert popup
    - Calender popup
    - child browser popup
    - Notification pop up
    - File Upload popup
11) Dropdown

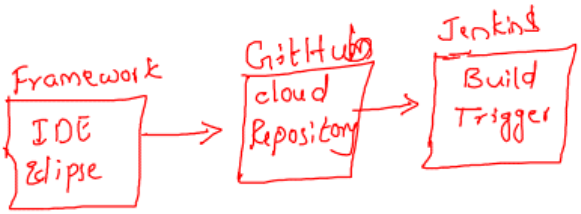
Chapter – 3

1) Data Driven Testing
   - Properties file
   - Excel file
2) POM – Page Object Model
   - Repository
   - POM
   - Stages
3) TestNG – Test Next Generation – tool
   - Download & install
   - Annotations
   - Priority & InvocationCount
   - Disable Testcase
   - Run testcases sequentially
   - Parallel execution
   - Re-run failed testcases
   - Assertions

Chapter – 4

Framework / Project

1) Maven Project
   - what, why, Adv
2) Dependencies & plugins
3) Design
4) Implementation
5) Execution
6) Report generations
7) GitHub
8) Jenkins



Java Concepts

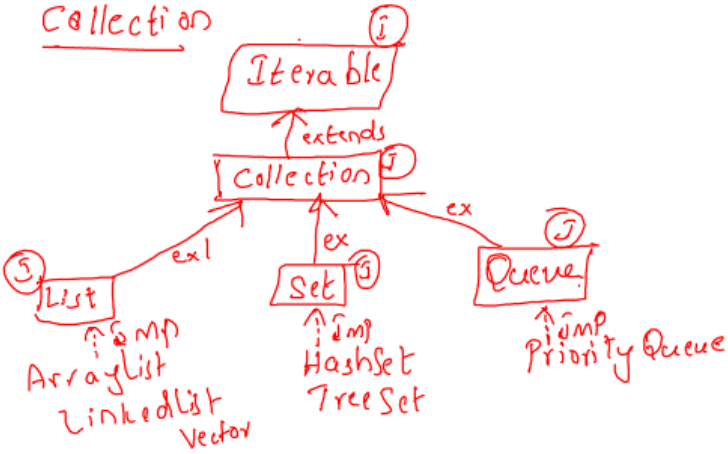
1) Token — keywords, identifiers, Literals, separators
2) Datatypes — Primitive →
   Non-primitive
3) Variables → Global → static
   → non static
   → Local
4) Typecasting — Primitive → Narrowing → Explicit
   → Widening → Implicit
   Non primitive → Upcasting
   → Downcasting
5) Decision stmts → if, if-else, else-if ladder, switch
6) control stmts → break, return, continue
7) loops → while, do-while, for, for each
8) Methods | functions →

OOPs

1) Encapsulation → data hiding → 'private'
2) Inheritance → 1) single level ✓
   2) Multilevel
   3) Multiple ✓ → Interface
   4) Hierarchical
   5) Hybrid
3) Polymorphism → ✓
   1) Compile time → method overloading
   2) Run time → method overriding
   → inheritance
4) Abstraction → class → upto 100%
   → interface → 100%

Exception → classes
- checked → compiler aware → try/catch or throws
- unchecked → try & catch & finally

Collection



Chapter-1

Introduction to Automation

what is Automation?

The process of converting any manual test cases into automation test scripts using Automation tool with programming or scripting language.

Programming Language :-

It is a language which is used to develop an application.

Ex: Java, C, C++, C#, Ruby, .net

Scripting Language :-

It is used to validate the application. Ex: Python, VB scripting, Javascript, Node JS, Perl

Advantages of Automation:

1) Saves time ✓
2) Accuracy in results ✓
3) Quality is good
4) Reusability of scripts ✓
5) Avoid repetitive tasks
6) Reduces human effort / Less resources
7) Test in detail

Disadvantages of Automation:

1) OTP
2) Captcha
3) QR code
4) Games
5) Audio & video test cases
6) printer related
7) Fingerprints
8) Embedded related
9) face-recognition
10) I'm not robot
11) knowledge of Programming is required

When we start Automation?

→ When application is functionally stable
→ Resources should be ready
  → Test cases
  → Tools
→ Long term projects
→ More repetitive tasks
→ More regression cycles ✓
  → Unit, Regional, Full regression
    ↳ manually

manually

10 ✓

100 →

Automation tools

1) Functional Automation tools
2) Non-functional Automation tools

1) Functional Automation tools –

Tools which are used to perform functional, integration, end to end test cases

Ex: Selenium, QTP, Winium, Appium, Selendroid, Rest Assured, Test Complete, Cypress etc

2) Non-functional Automation tools –

Tools which are used to perform performance related ~~ing~~ testing.

Ex: Load Runner, JMeter, Neo load, App load, No load etc

Introduction to Selenium

What is Selenium?

Selenium is an open source web application Test Automation Tool

Open source → source code is visible
Download & install w/o license
We customize it

Web Application → Any application rendered over web/internet
or
Any application which opens via browsers & work via internet

Test Automation — Testing any application without manual intervention

Tool → software

Why Selenium?

1) Open source → Download & install w/o license

Cost efficient

2) Open source → Integrate Selenium with third party tools

(*) Customization is easy

3) Platform Independent → installed on any OS (windows, Linux, IOS) (Operating System)

System compatibility testing is easy

4) It supports multiple programming languages

Java, C, C#, Python, Ruby, Javascript, Perl, R, Dart, Tcl, .net, PHP, Haskell, Elixir, objective C

5) It supports multiple browsers

Chrome, Firefox, Opera, IE, Safari, Edge

Browser Compatibility Testing easy

Drawbacks of Selenium ✓

1) It supports only web based applications

2) It needs programming knowledge

3) Automation disadvantages

Parts of Selenium

1) Selenium WebDriver
2) Selenium IDE (Integrated Development Environment)

Selenium IDE → If you want to create quick bug reproduction scripts.

→ We cannot perform click action

→ If you want to create scripts for exploratory testing (automation-aided)

→ It supports only chrome & firefox

Selenium WebDriver :

It is successor of Selenium RC (Remote Control) → old → deprecated

If you want to create robust, browser-based automation suite we go for Selenium WebDriver

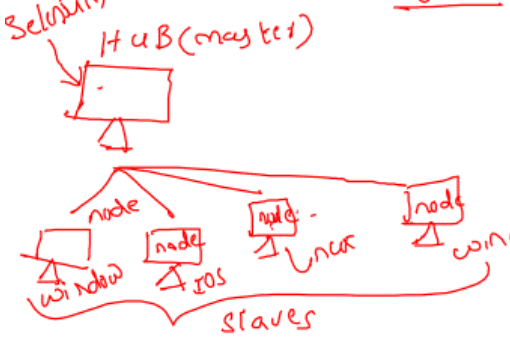
If you want to distribute scripts across many environments we go for Selenium WebDriver.

Development	Testing	UAT	Production

WBT

Selenium Grid → Selenium RC + Selenium WebDriver

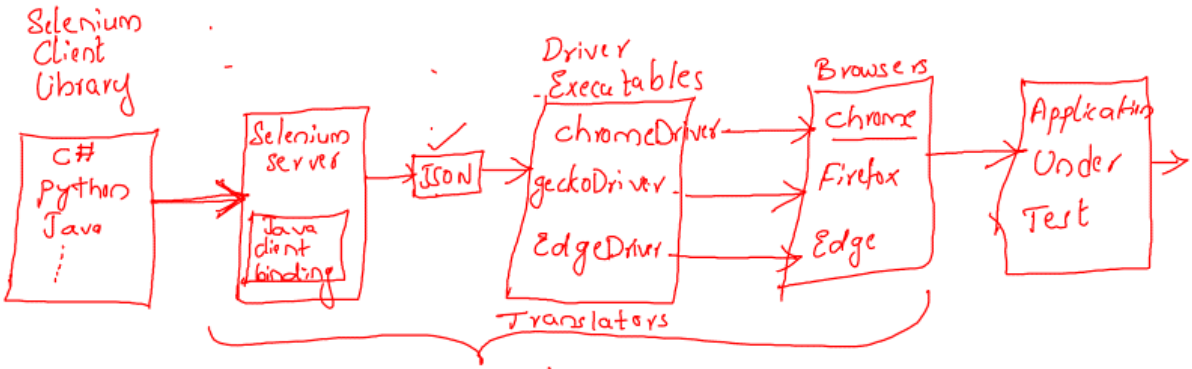
web apps → To perform System compatibility testing



To automate mobile based applications

1) Selendroid → Android
2) Appium → Android, IOS
3) Winium → windows

Selenium Architecture



W3C Protocol

World Wide Web Consortium Protocol

→ Selenium supports multiple programming languages which is called Selenium client library

→ Selenium internally has Java Client binding

→ When the code is triggered, then Selenium server converts the code to JSON format

JSON → Java Script Object Notation

→ JSON file is fed to driver executables

Note: Every browser has its own driver executables

Ex: Chrome browser → chromedriver.exe
Firefox Browser → ~~gecko~~ geckodriver.exe

→ Driver executables execute the commands on respective browsers and test the application and provide the result to client

client → program in our system

Softwares required for Selenium

1) JDK → 1.8 version
2) Eclipse IDE → 2022-09
3) Selenium Server
4) Driver executables
chromedriver.exe, geckodriver.exe, msedgedriver.exe

Steps to Download Selenium Server

1) Open the browser and type 'selenium.dev'
2) click on 'downloads'
3) click on selenium version link

Steps to download Driver executables

chrome driver.exe download

1) Check your chrome browser version
   1. Open chrome browser and click on 3 dots present at top right corner
   2. click on 'Help'
   3. click on 'About Google chrome' → Version - 107.0.5304.123

2) Open the browser and type 'selenium.dev'
3) click on 'Downloads'
4) Scroll the page till 'platforms Supported by Selenium'
5) click on 'Browsers +'
6) Under chrome → click on 'documentation'
7) click on 'Downloads'
8) Download respective driver executable

Download Edge & Firefox driver executables (optional) similarly

Steps to create Java Project in Eclipse

1) Open Eclipse & click on File → New → Java Project
2) Specify the name as 'SeleniumAutomation'
   (should not select module-info-java)
3) Click on 'Finish'..

ChromeDriver driver = new ChromeDriver();

This statement is used to launch empty chrome browser

ChromeDriver – It is a class

driver → It is a reference variable

= → Assignment operator

new → It is a keyword to create a new object

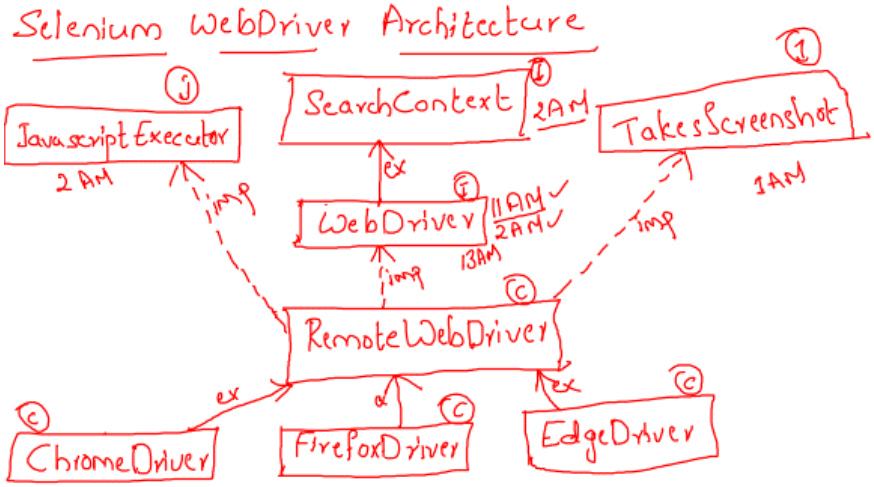
ChromeDriver() → It is a constructor of ChromeDriver class

Task1 → EdgeDriver & FirefoxDriver ✓

Task 2 → Write a script to launch multiple browsers.

Task 3 → Write a script to launch user desired browser

Selenium WebDriver Architecture



→ SearchContext is the supermost interface in Selenium WebDriver architecture. It has 2 abstract methods.

1> findElement() 2> findElements()

→ WebDriver is the sub-interface of SearchContext interface. It has 11 abstract methods of its own and inherits 2 abstract methods from SearchContext interface. All together it has 13 abstract methods

1> get()
2> getTitle()
3> getCurrentUrl()
4> getPageSource()
5> getWindowHandle()
6> getWindowHandles()
7> manage()
8> navigate()
9> switchTo()
10> close()
11> quit()

→ RemoteWebDriver is the implementing class of WebDriver interface

All the browser specific classes like

ChromeDriver for chrome

FirefoxDriver for firefox

EdgeDriver for edge

extend to RemoteWebDriver class

→ RemoteWebDriver class also implements to JavascriptExecutor interface and TakesScreenshot interface.

⇒ JavascriptExecutor has 2 abstract methods

1) executeScript()

2) executeAsyncScript

→ TakesScreenshot interface has one abstract method

1) getScreenshotAs()

WebDriver driver = new ChromeDriver();

what is upcasting?

The process of converting subclass to super type (class/interface) is called upcasting

WebDriver – interface

driver – It is reference variable

= → assignment operator

new → keyword to create an object

ChromeDriver() → constructor of ChromeDriver class

; → separator

WebDriver driver = new ChromeDriver();

Here we are creating an object for ChromeDriver class and upcasting it to WebDriver interface

→ It launches empty chrome browser

→ It enables us to access all the WebDriver methods

How to maximize browser?

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

How to minimize browser?

driver.manage().window().minimize();

How to make fullscreen?

driver.manage().window().fullscreen();

Few Web

Few WebDriver methods

1) get() → It is used to navigate to the application

Usage → driver.get(url); url - should be given in " "

2) getTitle() → It is used to return the title of Web Page

Usage → String title = driver.getTitle();

3) getCurrentUrl() → It is used to return the current Url

Usage → String url = driver.getCurrentUrl();

4) getPageSource() → It is used to return source code of webpage

Usage → String pagesource = driver.getPageSource();

5) close() → To close the current tab

Usage → driver.close(); (java.net.SocketException)

6) quit() → To exit the browser

Usage → driver.quit();

Task 1 → Write a script to launch chrome browser and navigate to amazon.com and fetch title, Url, Pagesource and close the browser

Task 2 → Repeat task1 for demo.actitime.com

Task 3 → Repeat task1 for flipkart.com

Navigation APIs

1) Back → driver.navigate().back();

2) After → driver.navigate().forward();

3) Refresh → driver.navigate().refresh();

4) To navigate to another application —
driver.navigate().to(url);

3) refresh – driver.navigate().refresh();
It is used to refresh the web page

4) to() → driver.navigate().to(url);
It is used to navigate to specified application.

HTML – Hyper Text Markup Language

HTML is used to develop web pages.
Web Designers, developers use this language.

HTML has 3 types of elements

1) Tags
2) Attributes
3) Text

1) Tag: Anything which starts from angular brackets is called a tag.

Ex: <input id="user" value = " " />
tag — closing the tag

<input> → input actions

<a> → anchor tag → identify links

<select> → dropdowns

Ex:
```
<select>
    <option>
    <option>
</select>
```
options in dropdown

<button> → button

<img> → images

 → body,

 → break

<html> → root

<head> → header

<title> → pagetitle

2) Attributes — Anything which is present in the form of key-value pairs inside '< >' is called attribute.

Ex: <input id="abc" name="user" />

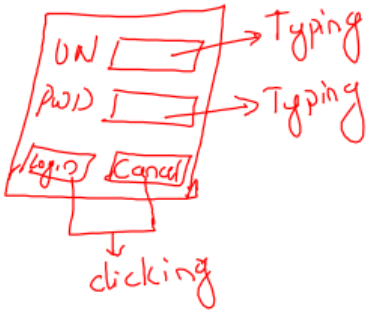
attributes

3) Text — Anything which is not enclosed in the angular brackets is called text

Ex: Login , Skillrary

text text

Locators



Why locators?

Inorder to check the existence of particular element on web page we use locators.

What are locators?

In selenium Locators are static methods of 'By' class.

'By' is an abstract class.

Types of locators

1) id
2) name
3) LinkText
4) partialLinkText
5) classname
6) tagname
7) cssSelector
8) xpath

1) id: If an element node has 'id' as an attribute then we go for id locator.

Usage:

```
driver.findElement(By.id("attributeValue"));
```

Scenario

1) Open the browser
2) Enter facebook.com
3) Type your name in username text field
4) close the browser

2. name locator: If an element node has an attribute as name then we go for name locator.

Usage: driver.findElement(By.name("name_attribute_Value"));

Scenario:

Open the browser and enter demo.actitime.com

Enter valid username and password credentials

Click on login button

Validate home page

Close browser

Task: Scenario:

Open the browser and enter facebook.com

Enter valid username and password

Click on loginbutton and validate home page

Close browser

3. linkText locator: It is used to identify links only.

How to identify links?

-> By look and feel

-> Check if the tagname is <a>

If we have text on the link then we go for linkText locator

Ex: <a>skillrary -> skillrary is the text on the link

Usage: driver.findElement(By.linkText("text_On_link"));

Scenario:

Open the browser and enter facebook.com

Click on Create a page

Close browser

Task: Scenario:

Open the browser and enter skillrary.com

Click on GEARS

Close the browser

Scenario:

Open the browser and enter facebook.com

click on Forgotten Password?

close the browser

4. partialLinkText: When the text on the link is lengthy then we go for partial link text

Ex: <a>Forgot your password?

partialLinkText is used for links only

Usage: driver.findElement(By.partialLinkText("partial text on link"));

Scenario:

Open the browser and enter demo.actitime.com

click on forgot your password link

close the browser

Task: Scenario: Open the browser and enter facebook.com

click on Forgotten Password

close browser

Scenario: Open the browser and enter demo.actitime.com

Click on Forgot your password and click on Return to login

Page and close browser

5. cssSelector locator: Whenever we don't have id or name attributes and whenever we don't have text on the link then we go for cssSelector.
CSS - Cascading Style Sheet

syntax - tagname[AttributeName = AttributeValue]
Usage: driver.findElement(By.cssSelector("cssSelector syntax expression"));
Steps to write cssSelector/xpath expression in html:
1. Right click on the element
2. Click on inspect
3. Press ctrl+f

5. cssSelector: tagname[attributeName = 'attributeValue']

get()	navigate()
It is used to navigate to an application and waits till the application is completely loaded	It is used to navigate to an application, it also navigates back, forward and refresh
It cannot store cookies	It stores all the cookies

6. xpath: To locate an element when the path of its node is unknown in html tree structure we go for xpath.
1. Absolute path : The complete path from the root of the html tree to particular element node is called absolute path. Here we use '/' to traverse.

```
<html>
  <head>
    <div>
      <input id='A' />          html/head/div[2]/input[1] - C
      <input id='B' />          html/head/div[1]/input[2] - B
    </div>                      html/head/div[1]/input[1] - A
    <div>                       html/head/div[2]/input[2] - D
      <input id='C' />
      <input id='D' />          html/head/div[2]/input[1]
    </div>
  </head>
</html>
```

Drawbacks: Absolute path can be lengthy
If we miss a single node the element cannot be identified
Time consuming to write the entire path

2. Relative path: The path from any parent node to particular element is called relative path. Here we use '//' for traversing.

```
<html>
  <head>
    <div>
      <input type='A' />      //div[1]/input[1] - A
      <input type='B' />      //div[1]/input[2]  - B
    </div>
    <div>
      <input type='C' />      //div[2]/input[1]  - C
      <input type='D' />      //div[2]/input[2]  - D
    </div>
  </head>
</html>
```

Relative xpath :

1. Basic relative xpath
2. Advanced relative xpath

1. Basic relative xpath:

1. xpath by attribute : Whenever we have attributes in the element node we go for xpath by attribute.

syntax: //tagname[@attributeName='attributeValue']

Scenario:
Open the browser
Enter demo.actitime.com
Enter username and password
click on login button
Validate home page
close the browser
Task : Write login script for facebook.com (use xpath by attribute)

Drawback: Attributes are mandatory and it doesnot support text.

2. xpath by text(): Whenever we have text on the element we go for xpath by text.

syntax: //tagname[text()='textValue']
//tagname[.='textValue']

Scenario:
Open the browser
Enter skillrary.com
Click on GEARS
Click on Skillrary Demo App
Close the browser

Scenario:
Open the browser
Enter demoapp.skillrary.com
Click on FEEDBACK
Close the browser

Drawback: Element should have text on it
Sometimes text can be lengthy
Doesnot support attributes

3. xpath by contains(): Whenever we have lengthy texts or lengthy attributes we go for xpath by contains().

-> Handles lengthy texts and attributes
-> Handles spaces
-> Handles partially changing elements
-> Handles non-breakable spaces

syntax: //tagname[contains(@attributeName,'attributeValue')]
//tagname[contains(text(),'textValue')]

Scenario:
Open the browser
Enter facebook.com
Click on forgotten password link
Type mobile number and click on search
Close browser

Scenario:
Open the browser
Enter demo.actitime.com
Enter valid credentials and login
Click on '?' and go to About your actitime
Inspect the build version element
Inspect number of user accounts
Inspect Product version

Whenever developer/web designer develops a web page sometimes they give non breakables spaces in the text of an element using .
We cannot identify or locate such elements using xpath by text()
We should use xpath by contains().

Scenario: Open the browser and type demo.vtiger.com
Inspect Products, Features, Solutions, Take Product tour, Resources

2. Advanced relative xpath:

1. xpath by group index: When we have more than one matching elements we go for xpath by group index.

syntax: (xpath exression)[PositionValue]

Drawbacks:
- There can be n number of matching elements and indexing by searching each and every element is a tedious job.
- With frequent GUI changes, the elements position might not remain same. Hence we cannot locate the element uniquely.

2. xpath by traversing:

Steps to be followed:
1. Identify static element and write the xpath
2. Identify common parent
3. Write tagname of dynamic element and element index(if required)

We have two types in xpath by traversing
1. Independent and dependent xpath
2. xpath by axes

1. Independent and dependent xpath:

1. Forward traversing: Traversing from parent to immediate child using '/' is called forward traversing.
2. Backward traversing: Traversing from child to immediate parent using '/..' is called backward traversing.

2. Xpath by axes:

1. parent axes: It is used to traverse to immediate parent. It works like '/..'

   Usage: /parent::tagname

2. child axes: It is used to traverse to immediate child. It works like '/'

   Usage: /child::tagname

3. ancestor axes: It is used to traverse from child to any parent node.

   Usage: /ancestor::tagname

4. descendant axes: It is used to traverse to any child from parent node

   Usage: /descendant::tagname

5. Sibling functions:

1.preceding-sibling: It is used to traverse to the nodes above which are present at the same level having same parent.

   Usage: /preceding-sibling::tagname

2. following-sibling: It is used to traverse to the nodes below which are present at the same level having same parent.

   Usage: /following-sibling::tagname

6. preceding: It is used to traverse to the nodes at the same level above having different parents.

   Usage: /preceding::tagname

7. following: It is used to traverse to the nodes at the same level below having different parents.

   Usage: /following::tagname

Operators in xpath:

We have two operators to pass multiple attributes to locate an element uniquely

1. and : When both conditions satisfy only then it locates the element

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

   or

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

   or

   //tagname[contains() and contains()]

2. or : When atleast one condition is satisfied only then it locates the element s

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

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

Differences between cssSelector and xpath

cssSelector	xpath
cssSelector is faster compared to xpath	xpath is bit slower
It is unidirectional	It is multi-directional
We cannot use multiple attributes	We can use multiple attributes using 'and' and 'or' operators
It doesnot support text and indexing	It supports text and indexing

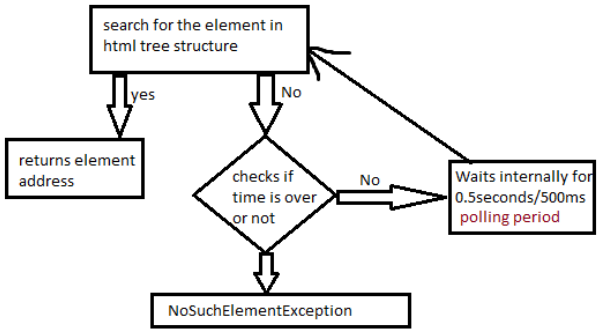
7. className locator: Whenever we have an attribute as class in the element node we go for className locator.
Usage: driver.findElement(By.className("class_attribute_value"));
Drawback: We might have multiple matching elements using same classname, hence we cannot identify the element uniquely

8. tagName locator: It is used to identify an element using tagname.
Usage: driver.findElement(By.tagName("tagName"));
We can also fetch the list of elements having same tagname.
Drawback: There can be multiple matching elements using same tagname, hence we cannot identify an element uniquely.

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

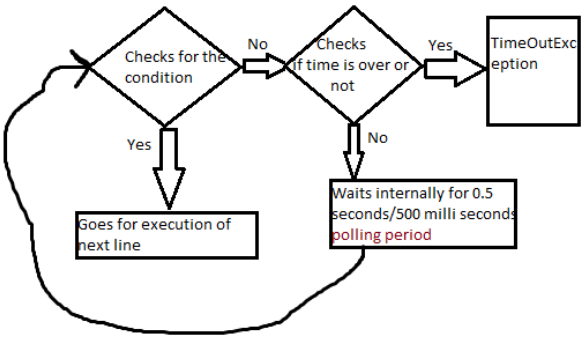
Different Wait statements:
1. Thread.sleep(time_in_milliseconds); -> It is simple java wait statement. It blindly waits for specified amount of time and continues executing next statements. It throws InterruptedException.

2. ImplicitlyWait statement : It is selenium wait statement which synchronizes only findElement() and findElements() methods.
Syntax: driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(time_in_seconds));



ImplicitlyWait WorkFlow:
-> Whenever implicitlyWait statement is given along with findElement() or findElements() methods, it searches for the element in html tree structure.
-> If element is not found, then it checks if given time is over or not
-> If given time is not over, then it internally waits for 0.5 seconds or 500 milli seconds. This waiting time is called as polling period. Then it goes back and searches for the element in html tree structure again.
-> If time is over then it throws NoSuchElementException
-> If element is found it returns element address.
-> This process repeats until element is found or specified time is over.

3. ExplicitlyWait: It is selenium wiat statement which synchronizes all the WebDriver methods including findElement() and findElements() methods.
Syntax:
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(timeInSec));
wait.until(ExpectedConditions.visibilityOf(element));
or
wait.until(ExpectedConditions.elementToBeClickable(element));
or
wait.until(ExpectedConditions.titleContains("title"));



ExplicitlyWait Workflow:
-> Whenever explicitlyWait statement is given along with any WebDriver methods, it will first check if the condition mentioned is satisfied or not.
-> If the condition is satisfied it will go to the next statement in the script.
-> If the condition is not satisfied, then it checks if specified time is over or not.
-> If time is over, it throws TimeOutException
-> If time is not over , it will wait internally for 0.5seconds or 500 milliseconds. This waiting period is called polling period. Then it re-checks if the condition is satisfied or not.
-> This process repeats until condition is satisfied or time is over.

ImplicitlyWait	ExplicitlyWait
It synchronizes only findElement() and findElements() methods	It synchronizes all the WebDriver methods including findElement() and findElements()
No need to specify condition	Condition should be specified
Once implicitlyWait is given it works implicitly, need not give again	ExplicitlyWait should be given every time for method to synchronize it
It throws NoSuchElementException if element is not found	It throws TimeOutException if condition is not satisfied

4. FluentWait: It is selenium wait statement which is used to customize polling period.

Syntax:

```
FluentWait wait = new FluentWait(driver)
                .withTimeout(Duration.ofSeconds(time))
                .pollingEvery(Duration.ofSeconds(polling_time))
                .ignoring(Exception e);
```

WebElement: It is an interface.

Element which appears on the web page are called web elements.

WebElement methods:

Actions	Getters	Verification
sendKeys()	getText()	isEnabled()
click()	getLocation()	isDisplayed()
clear()	getSize()	isSelected()
submit()	getAttribute()	

Actions:

1. sendKeys() - It is used for typing action.
2. click() - It is used for clicking
3. clear() - It is used to clear data in text field

   Scenario: Open the browser and enter google.com,
   Type some text in search textfield and clear the data
   Close the browser

   Task: Repeat same scenario for amazon.com

4. submit() - It is similar to click method. But it is used for forms and it works only if the element node has the attribute type="submit"

   Scenario: Open the browser and enter amazon.com
   Type dresses in search text field and click on search button
   Close browser

WebElement methods:

Getters:

1. getText(): It is used to fetch text on the element.
   Returntype is String.

   Scenario: Open the browser and enter demo.actitime.com
   Fetch the page header text

2. getLocation(): It is used to fetch the location of the element on web page
   Returntype is Point
   Point is a class in selenium which provides the x and y coordinates of the element
   For x coordinate -> getX() For y coordinate -> getY()

Priority for locators:

id -------> name --------> linkText ----------> xpath
(only if it is link)

3. getSize(): It is used to get size or dimensions of the element on the web page.
Return type of getSize() is Dimension.
Dimension is a class in selenium. It has two methods to get height and width of the element.
Height -> getHeight()
Width -> getWidth()

Scenario:
Open the browser and enter amazon.com
Fetch dimensions of any element on the web page
close the browser
Scenario:
Open the browser and enter facebook.com
Fetch the dimensions of 'facebook'
close the browser

4. getAttribute(String attributeName): It is used to fetch attribute value of the element when attribute name is passed as key to this method.
Returntype is String.

Scenario:
Open the browser and enter google.com
Fetch the attribute of search text field
close the browser

Verification methods:

1. isDisplayed(): It is used to check if the element is displayed on the web page or not
Returntype is boolean.

Scenario:
Open the browser and enter demo.vtiger.com
Check if the vtiger logo is displayed or not
Close the browser
Scenario: Open the browser and enter demo.actitime.com, check if the logo is displayed or not and close the browser

2. isEnabled(): It is used to check if the element on the web page is enabled or not
Returntype is boolean

Scenario:
Open the browser and enter the url amazon.com
Check if the search button is enabled or not
Close the browser
Scenario:
Open the browser and enter instagram.com
Check if login button is enabled or not
Enter valid credentials
Check if login button is enabled or not
If enabled click on the button else print disabled
Close the browser.

3. isSelected(): It is used to check if the element is selected or not. Generally used for checkboxs and radio buttons.
Return type is boolean.

Scenario:
Open the browser and enter demo.actime.com
Select the checkbox 'Keep me logged in'
Check if the checkbox is selected or not
close the browser

Scenario:
Open the browser and enter facebook.com
Click on 'Create New Account'
Select gender
Check if gender radio button is selected or not
Close the browser

Interview Questions

1. Explain Selenium Architecture.
2. Explain WebDriver driver = new ChromeDriver(); statement.
3. Explain Selenium WebDriver Architecture/WebDriver class diagram/java Selenium architecture.
4. Explain how to manage window.
5. Explain navigation APIs
6. What are the different ways to navigate to an application?
7. Differences between get and navigate
8. Explain locators-> why? what? types? brief description
9. Explain xpath
10. Differences between cssSelector and xpath
11. Explain synchronization in Selenium.
why? what? Types of waits and Brief description with syntaxes and worflow
12. What is polling period ? Can you customize it?
13. Differences between implicitlyWait and explicitlyWait
14. What is WebElement? Explain the WebElement methods.
15. Difference between click() and submit()
16. What are the different ways to validate a web page?
17. Write login script.

Chapter - 2
Handling WebElements

1. Auto Suggestions: List of suggestions that appear automatically on the web page when we search a resource.
We can handle auto suggestions using findElements() method.

Scenario:
Open the browser and enter google.com
Type 'selenium' in the text field
Fetch all the auto suggestions and print in console
Close the browser
Scenario:
Open the browser and enter amazon.com
Type 'laptops' in search text field
Fetch all the auto suggections and print in the console
Close the browser

findElements() : It is used to fetch the list of matching web elements on the web page
Return type of findElements() is List<WebElement>

Difference between findElement() and findElements()

findElement()	findElements()
It is used to fetch first matching element	It is used to fetch the list of all matching elements
The return type is WebElement	The return type is List<WebElement>
If the element is not found it throws NoSuchElementException	If the elements are not found it returns empty array list

Task: Scenario:
Open the browser and enter google.com
Type your name in search text field
Fetch the 5th element in autosuggestions
Close the browser

Mouse Actions: Mouse actions can be handled using Actions class.
Different mouse actions:
1. Mouse Hovering
2. Right click
3. Double click
4. Drag and drop

Step 1: Create an instance for Actions class and pass WebDriver reference to the constructor
Actions a = new Actions(driver);
Step 2: Call respective methods to perform mouse actions using Actions class reference
Mouse Hover: a.moveToElement(element).perform();
Right Click: a.contextClick(element).perform();
Double Click: a.doubleClick(element).perform();
Drag and Drop : a.dragAndDrop(src,target).perform();
perform() is the default method to be given after each mouse action method

Scenario:
Open the browser and enter demoapp.skillrary.com
Mouse Hover to the course
Click on Cucumber tab
close the browser

Note: When the element is not inspectable by right click and vanishes within seconds even before you perform inspection, follow the steps given below:
1. Go to sources in developer tools
2. Mouse Hover to the element
3. Press f8+ctrl+\
This will pause the script and puts the screen in debugger mode
4. Take the arrow mark to the element to inspect

2. Right click:
```
Actions a = new Actions(driver);
a.contextClick(element).perform();
```
Scenario:
Open the browser
Enter amazon.com
Right click on search text field
close the browser

Scenario: Open the browser
Enter myntra.com
Right click on Beauty tab
close the browser

3. Double click:
```
Actions a = new Actions(driver);
a.doubleClick(element).perform();
```
Scenario:
Open the browser
Enter demoapp.skillrary.com
Mouse Hover to course
Click on 'Selenium Training'
Double click on '+' button
close the browser

4. Drag and Drop:
```
Actions a = new Actions(driver);
a.dragAndDrop(src,target).perform();
```
Scenario: Open the browser and enter given url, drag and drop cat to first box, close the browser.

Drop downs:
We can handle dropdowns using 'Select' class.
Select class is present in org.openqa.selenium.support package
We have three methods to perform select from dropdown ->
1. selectByIndex(int index)
2. selectByVisibleText(String text)
3. selectByValue(String value)

Steps to handle dropdowns:
1. Create an instance of Select class and pass element reference as argument to the constructor
   Select s = new Select(element);
2. Using the Select class reference call one of the above methods to select an element from dropdown list

Note: Dropdowns can be identified using the tagname <select>

Scenario:
Open the browser and enter amazon.com
Select an item from All dropdown
Close the browser

To get the first selected option - getFirstSelectedOption()
This method returns the WebElement which is selected first.

To get all the options from dropdown list - getOptions()
This method returns List<WebElement>

We have two types of dropdowns:
1. single select dropdown- We can select only one option at a time. We cannot deselect the single select dropdown. It throws UnsupportedOperationException
2. multi select dropdown- We can select multiple options at a time. We can deselect from multi select dropdown.

To check if the dropdown is single select or multi select we use the method - isMultiple()
It returns true if the dropdown is multi select
It returns false if the dropdown is single select

Scenario:
Open the browser
Enter demoapp.skillrary.com
Check if the dropdown is single select or multi select
close the browser

In order to deselect the options from the dropdown:
1. deselectByIndex(int index)
2. deselectByVisibleText(String text)
3. deselectByValue(String value)
4. deselectAll()

Scenario: Open the browser
Enter demoapp.skillrary.com
Select first three options
Get all selected options
Deselect the options
Close the browser

To get all selected options - getAllSelectedOptions()
This method returns List<WebElement>

Screenshot:
To get the screenshot of the web page we use TakesScreenshot interface and we should add 'apache commons io' libraries to the project.

Screenshot:
Pre requisite: Download apache common io library and add to project's build path

Steps to acheive Screenshot of webpage:
1. Typecast WebDriver reference to TakesScreenshot interface
   TakesScreenshot ts = (TakesScreenshot) driver;
2. With TakesScreenshot reference call the method getScreenshotAs()
   File src = ts.getScreenshotAs(OutputType.FILE);
3. Create a new File in project
   File dest = new File("./screenshot/screenshot.png");
4. Copy the src file(temporary file) to dest file(permanent file)
   FileUtils.copyFile(src,dest);
The above statement throws IOException
Note: In order to copy the Screenshot file from temporary memory(RAM) to permanent memory(Local disk) we have to use apache commons io libraries.
FileUtils is the class provided by apache commons io libraries, using this class we call a method copyFile() to copy the file from RAM to permanent memory

5. After execution refresh the project and you can see screenshot folder created

Task:
Scenario: Open the browser
Enter demo.actitime.com
Enter wrong credentials and click on login
Takes the screenshot
close the browser.

case 1: Hard coding the coordinates
```
js.executeScript("window.scrollBy(0,5000)");
```
Scenario:
Open the browser
Enter myntra.com
Scroll the page
Close the browser

Task: Repeat above scenario for amazon.com, ebay.com
case 2: Using the location of the element and scroll till that point
1. Find the location of the element
2. Concatenate the x and the y coordinates in the above script
```
js.executeScript("window.scrollBy("+x+","+y+")");
```
Scenario:
Open the browser
Enter myntra.com
Scroll the page till sarees element
Close the browser

Task:
Repeat above scenario for amazon.com

case 3: Using element reference
```
js.executeScript("arguments[0].scrollIntoView(true)",elementReference);
```

Frames: The web page inside another web page is called frame.
To handle frames we should switch the control to the frame.

To switch to the frame:
driver.switchTo().frame(index);
driver.switchTo().frame(id_or_name);
driver.switchTo().frame(text);

To switch back to from the frame
driver.switchTo().defaultContent();

Scenario:
Open the browser
Enter snapdeal.com
Mouse Hover on sign in and click login
Then enter mobile number and click on login
close browser

Popups: Popups are the windows which appear on the screen.
Different types of popups.
1. Alert popup:
Not inspectable and not movable
To handle this popups first we have to switch the control to the popup window.
To click on ok -> driver.switchTo().alert().accept();
To click on cancel -> driver.switchTo().alert().dismiss();
To get text on the popup -> driver.switchTo().alert().getText();
To pass data to the popup -> driver.switchTo().alert().sendKeys("data");

2. Hidden Division/Calender popup:
Inspectable but not movable
We can handle this popup using findElement()
Scenario: Open the browser, enter makemytrip.com and select departure date and click on search, close browser.

3. Notification popup:
Not inspectable and not movable.
To disable this pop up we have browser specific classes like ChromeOptions for chrome and FirefoxOptions for firefox.

Step 1: Create a reference to ChromeOptions class even before launching the browser.
ChromeOptions option = new ChromeOptions();
Step 2: Call the method addArguments() using ChromeOptions reference
option.addArguments("--disable-notifications");
Step 3: While launching browser call parameterized constructor with ChromeOptions class reference as argument
WebDriver driver = new ChromeDriver(option);
Scenario:
Open the browser
Enter yatra.com
Handle notification popup
Close the browser

Child browser popup:
We handle this pop up using 2 methods:
1. getWindowHandle() - This method returns parent window address
Returntype is String
2. getWindowHandles() - This method returns both parent and child browser addresses
Returntype is Set<String>

Scenario:
Open the browser
Enter skillrary.com
Click on 'GEARS'
Click on 'SKILLRARY ESSAY'
Type your name and click on 'Yes its my name'
Close the browser

File upload popup:
We can handle this pop up in three ways.
1. Using sendKeys()
2. AutoIT tool
3. Robot class

1. Using sendKeys(): We use sendKeys() to upload file only if we have type="file" attribute.
We should pass the path of the file to the sendKeys() method.

AutoIT: It is third party tool which automates StandAlone applications.
Steps to download AutoIT:
1. Open the browser and type autoIT download
2. Click on the first link
3. Scroll the page bit down and click on 'Download autoIT'
4. After downloading, double click on the file and install

Scenario:
Open the browser
Enter the naukri.com
Click on Register button
Click on Upload Resume
Close the browser

Selenium
eclipse IDE

Auto IT
SciTE Script Editor

Steps to launch SciTE Script Editor:
1. Type Scite Script Editor in search
2. Open the app

Standalone application Scenario steps:
1. Switch to the opened file popup window
2. Switch the control to Filename - text bar
3. Type file path
4. Click on Open button

Step 1: WinWaitActive("title")
This method is used to wait until the file upload popup window is active and switches the control to it.
Step 2: Sleep(2000)
In autoIT Sleep() is the only wait statement
Step 3: ControlFocus("title", "text","controlID")
In order to inspect an element in stand alone application
1. Search for 'autoIT v3 Window info' in computer
2. Drag and drop the 'Finder Tool' to the element to be inspected

File upload using Robot class:
Robot class is used to automate all keyboard related actions.
It is available in java.awt package
awt - Abstract Window ToolKit

Step 1: Create an instance for Robot class
Robot robot = new Robot();
Step 2: Create an instance for StringSelection class and pass the path of the file to be copied to the file upload popup.
StringSelection path = new StringSelection("file_path");
StringSelection class is available in java.awt.datatransfer package
It is used for all data transfer purposes
Step 3: Set the contents to the clipboard ucing ToolKit class.
ToolKit.getDefaultToolKit().getSystemClipboard().setContents(path,null);
Step 4: Press ctrl+v to copy file path from the clipboard
robot.keyPress(KeyEvent.VK_CONTROL);
robot.keyPress(KeyEvent.VK_V);
Step 5: Release the pressed keys
robot.keyRelease(KeyEvent.VK_CONTROL);
robot.keyRelease(KeyEvent.VK_V);
Step 6: Press and release enter button
robot.keyPress(KeyEvent.VK_ENTER);
robot.keyRelease(KeyEvent.VK_ENTER);

Windows:
Open a new tab and switch the control to it
driver.switchTo().newWindow(WindowType.TAB);

Open new window and switch the control to it
driver.switchTo().newWindow(WindowType.WINDOW);

Click on Control in AutoIT v3 Window Info

ControlFocus method is used to switch the control over to the element.
controlID : It is the combination of Class and Instance

Step 4: Send("path_of_the_file_to_be_uploaded")
Send method is used to type data in to the element
Step 5: Sleep(2000)
Step 6: ControlClick("title","text","controlID")
ControlClick method is used to click on the element

-> After writing the program, save it in a folder in desktop with .au3 extension.
-> To compile the program in Scite Script Editor, click on 'Tools' and click on 'Compile'
Click on 'Compile Script' button
We can see .exe file auto-saved to the folder on desktop

Now integrate the above code with Selenium program:
Runtime.getRunTime().exec("path of .exe file");

Handling WebElements

1. Auto Suggestions -> findElements() -> Google Scenario
2. Differences between findElement() and findElements()
3. How do you handle mouse actions?
4. How do you handle drop downs?
5. How do you get screenshot?
6. How do you handle scroll bar?
7. How do you handle frames?
8. How do you handle windows? Set<String>
9. Alert popup?
10. Calender popup?
11. Notification popup? Timeouts
12. Child browser pop up?
13. Differences between getWindowHandle() and getWindowHandles()
14. File upload popup? Navigation

Chapter 3

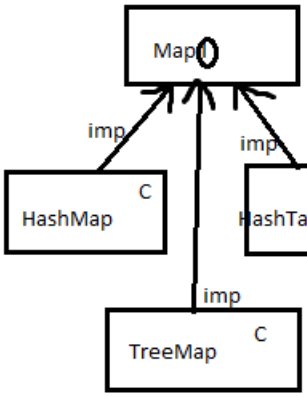
1. Data Driven Testing:
The process of driving the data from external resources like properties file or excel file and utilising it in the test scripts and performing test execution is called data driven testing.

Types of data:
1. Common data: The data which is common to all the test scripts is called common data.
Ex: url, login credentials and configuration settings
2. Test data: The data which is specific to the test script is called test data.

Properties file:
The data is stored in the form of key-value pairs in properties file
Ex: browser=chrome
username=admin
All the values stored will be in String format
time=10

Pre-requisite:
Create a properties file in the project
Steps:
Right click on the project -> New -> file
Name the file with '.properties' extension
Click on finish

Steps to read data from properties file:
1. Convert physical file into java readable object
FileInputStream fis = new FileInputStream("file path")
The above statement throws FileNotFoundException
2. Create an instance for Properties class
Properties property = new Properties();
Properties class is present in java.util package
3. Load all the key value pairs to Properties object
property.load(fis);
The above statement throws an exception - IOException
4. Read data from Properties file
String data = property.getProperty("key");

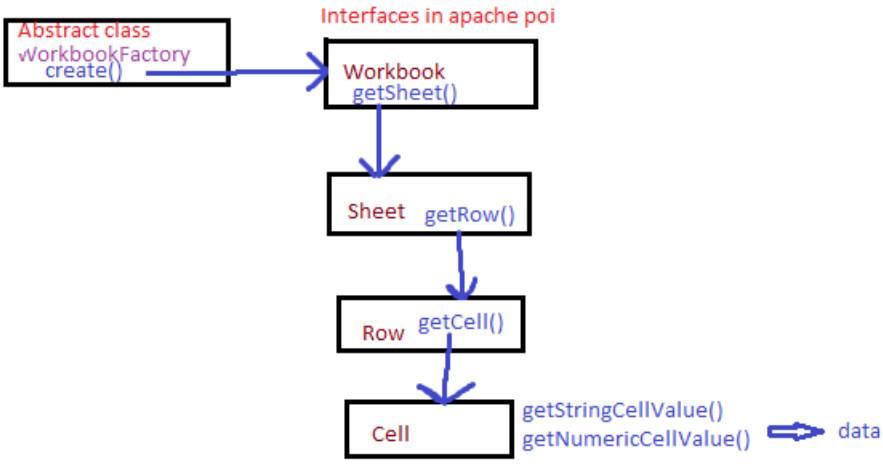


It stores data in the form of key-value pairs.
It doesnot allow duplicate keys.
It allows duplicate values.
It has get() method to fetch the data.
It has put() method to write data.

Java concept used internally in Properties file:
Internally Properties file utilizes Map interface concept to load data to Properties object.
When property.load(fis) method is called, it internally creates a HashTable and stores all the data from fis into HashTable in the form of key-value pairs.
Hash Table is the implementing class of Map interface.

Drawbacks of properties file:
1. We can fetch only single data at a time.
2. It doesnot allow duplicate keys
3. It is not organized.
4. When we have lot of data it is tedious to fetch particular key from the file.

Excel:
In excel the data is stored in the form of tables.
It is organized.
We store test data in excel.
We use apache poi libraries to read data from excel.



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

POM(Page Object Model): It is java design pattern preferred by Google to develop an object repository.

Advantages of POM:
1. Handles StaleElementReferenceException
2. Maintenance of web elements is easier.
3. Modification of web elements is easy.
4. Code can be optimized.
5. Reusability of elements and business libraries.
6. Faster test script development.
7. Increased code readability.

POM has 3 steps:
1. Declaration:
We declare the element as private.

```
@FindBy(LocatorName="LocatorValue")
private WebElement/List<WebElement> elementRef;
```

- @FindBy is used in POM to find element/elements as well as declare the element/ elements.
- It returns WebElement/ List<WebElement> based on the element declared.

2. Initialization:
Here we call parameterized constructor

```
public ClassName(WebDriver driver)
{
  PageFactory.initElements(driver,this);
}
```

3. Utilization: Here we give methods for the declared elements i.e., business libraries

StaleElementReferenceException:
It is selenium exception which occurs when we try to fetch an element using same old address.
Whenever the web page is reloaded its element references changes and when we try to fetch the element with same old reference then it throws StaleElementReferenceException.

In POM the above exception is handled by the constructor in the second step.
When the web page gets reloaded, the current addresses are reinitialized to the driver which avoids the occurence of the above exception.

Java Concept used in POM:
Encapsulation technic is used in POM.
Encapsulation is the process of binding states and behaviours of an object in a class. Data hiding is acheived in Encapsulation since we declare states of the object using private keyword.
In POM we declare the elements as private acheiving data hiding and access these elements using the respective business libraries in test script.

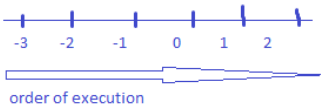
TestNG: Test Next Generation
- It is unit testing framework tool used by both developers and automation engineers.
- Developers use it for White Box Testing
- Automation Engineers use it for batch, group, parallel executions of test scripts.
- testNG is developed as a plugin for IDE(Integrated Development Environment)

We have other framework tools.
1. java - JUnit
2. .net - NUnit
3. Javascript - Jasmine, Mocha
4. Python - Pydev
- testNG is developed with combination of features of JUnit and NUnit and also has additional features.

Advantages of testNG:
1. It is open source tool - Downloading and installation is easy.
2. Set priority to the test scripts
3. Run same test script multiple times using InvocationCount
4. Disable test script
5. Batch Execution
6. Parallel Execution
   - Distributed Parallel
   - Cross Browser parallel/Browser compatibility testing
7. Group Execution
8. Re-run the failed test scripts
9. Generates html reports automatically
10. Assertions - Validation of test scripts
11. Create dependency between test scripts - dependsOnMethods
12. Annotations are used for controlled flow of test execution

- @Test acts like main method in Java.
- Execution in testNG starts from @Test
- Ideally we can have 15 @Test methods in a class

1. Prioritizing the test scripts:
To prioritize test scripts we have a parameter priority = int
Usage : @Test(priority = int)
- Default priority is 0
- Priority follows number line order



If the priority of the @Test methods is same then it will execute in the order of ASCII values of the method names.

2. Invocation count:
In order to run same test method multiple times with same data we use parameter invocationCount = int
Default invocationCount = 1

Usage: @Test(invocationCount = int)

If invocationCount is 0 or negative value, that @Test method will not be considered for execution.

3. Disabling the test script:
To disable the test script we use parameter enabled = false
By default enabled = true
Usage : @Test(enabled=false)

Steps to convert class to testng.xml file:
1. Select the class file and right click on it.
2. Go to TestNG and click on 'Convert to TestNG'
3. Rename the xml file with .xml extension and click on 'Finish'
   xml file will be created in the project
4. Double click on the file to open it
xml is the advanced level of html language
Comments - <!-- comment -->

Annotations in testng:

1. @BeforeSuite : It executes before the <suite> tag in xml.
2. @BeforeTest : It executes before <test> tag in xml.
3. @BeforeClass : It executes before <class> tag in xml.
4. @BeforeMethod : It executes before @Test method in a class.
5. @AfterMethod : It executes after @Test method in a class.
6. @AfterClass : It executes after </class> tag in xml.
7. @AfterTest : It executes after </test> tag in xml.
8. @AfterSuite : It executes after </suite> tag in xml.

```
<!-- BeforeSuite -->
<suite name="Suite">
  <!-- BeforeTest -->
  <test thread-count="5" name="Test">
    <classes>
      <!-- BeforeClass -->
      <class name="testNG.EnabledFalsePractice"/>
      <!-- AfterClass -->
    </classes>
  </test> <!-- Test -->
  <!-- AfterTest -->
</suite> <!-- Suite -->
<!-- AfterSuite -->
```

@BeforeMethod
@Test
@AfterMethod

Batch Execution:
Step 1: Select all the class files which are to be run in batch.
Step 2: Right click on the selected files and convert to testng.xml file.
Step 3: Run the .xml file

Reports:
TestNG has a special feature to generate html reports automatically when .xml file is run.

Step 1: Run the xml file
Step 2: Click on test output folder in the project
Step 3: Right click on emailable-report.html -> Open With -> Internal Browser
HTML report will be opened

Re-run failed test scripts:
When the test scripts fail, testng creates separate .xml file for these failed test scripts.

Step 1: Refresh the project
Step 2: Expand test output folder
Step 3: Open test-failed.xml file
        You can find the methods and classes that are failed
Step 4: Fix the defects and re-run test-failed.xml file

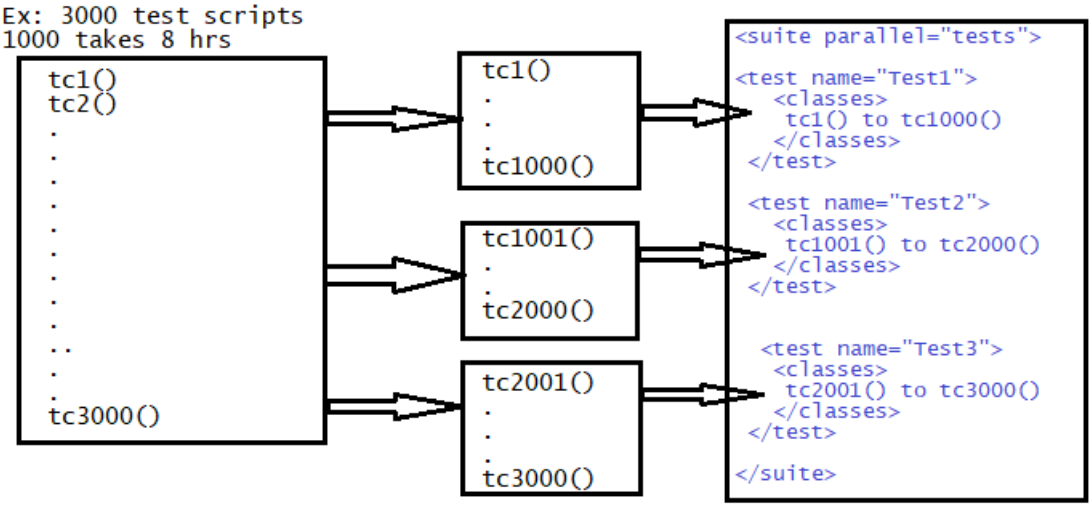
Group Execution:
In order to execute specific group of test scripts testNG has the feature "groups".

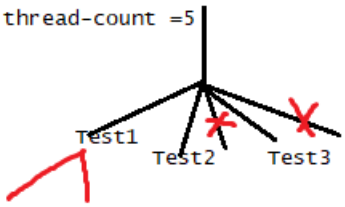
Step 1: We should specify the group for each @Test method as parameter.
Step 2: Convert all the class files to .xml file
Step 3: Mention the group to be run after suite tag

```
<groups>
  <run>
      <include>
      <exclude>
  </run>
</groups>
```

Parallel Execution:
Execution the test scripts in parallel
1. Distributed Parallel Execution:
Distributing the test scripts to different test runners and executing all the test runners in parallel is called distributed parallel execution.

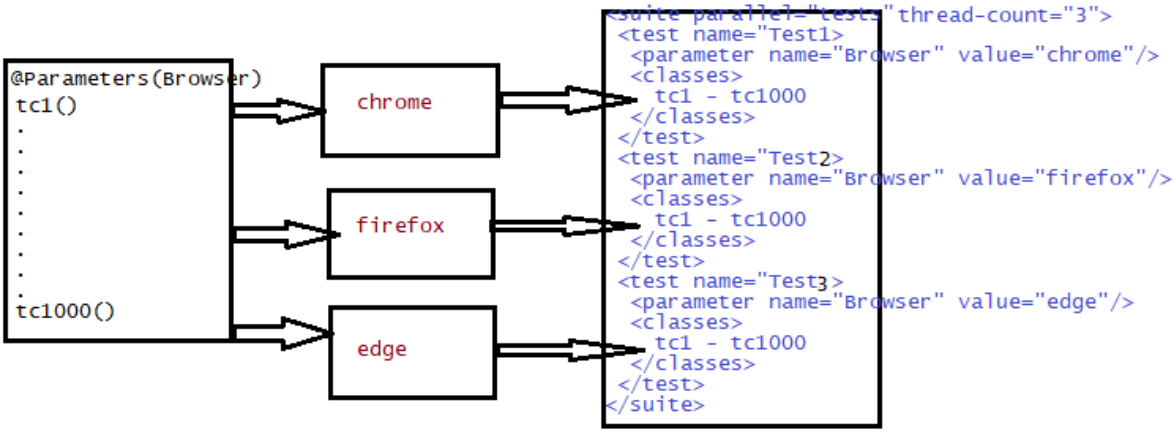
Ex: 3000 test scripts
1000 takes 8 hrs





Note: Thread-count should be equal to number of test runners
Default thread-count is 5
There is no maximum limit for thread-count

2. Cross browser parallel/ compatibility testing:
Executing same set of test scripts on multiple browsers is called cross browser parallel execution.

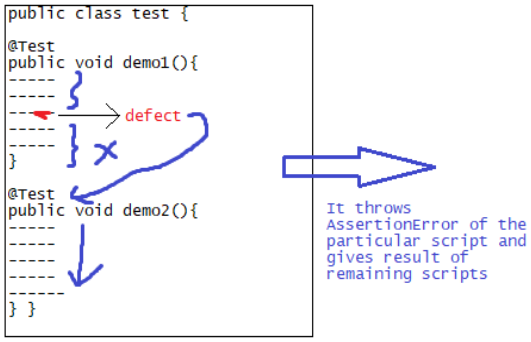


Assertions: It is testng feature which is used to validation or verification in the test scripts.

Why Assertions?
Normal if-else statement donot have the capability to fail the test script since depending on the condition mentioned it will either execute if-block or else-block but never fails the test script. That's why we use Assertions to do accurate validations in test script.

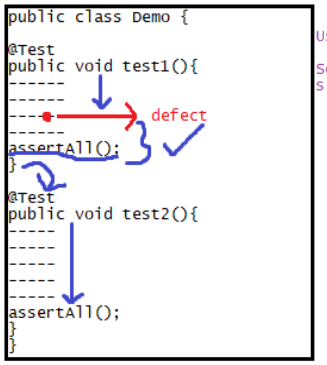
We have two types in Assertions:
1. Hard Assert/Assert
2. SoftAssert

1. Hard Assert/ Assert:
Assert is the testng class which has static methods to perform validations in test script.
1. assertEquals(actualValue, expectedValue)
2. assertNotEquals(actual, expected)
3. assertTrue(condition)
4. assertFalse(condition)
5. fail() - to fail the test script
Usage: Assert.assertTrue(condition);



When Assert is given it starts with normal execution of test script. When error occurs in one test block it throws AssertionError and skips the execution of remaining statements in the current block and transfers the control to next block. Finally it gives the result of execution of all the test blocks.

2. SoftAssert:
It is a class in testng which has all non-static methods to perform validations in test scripts.
1. assertEquals(actual, expected)
2. assertNotEquals(actual, expected)
3. assertTrue(condition)
4. assertFalse(condition)
5. assertAll() - mandatory method in SoftAssert and it should be given at the end of the block.



Usage:

```
SoftAssert s = new SoftAssert();
s.assertTrue(condition);
```

When SoftAssert is given, it starts with normal execution of test scripts. When error occurs in particular block it will still execute the remaining statements in the block and then transfer the control to the next block.
At the end it will throw AssertionError of the particular defect and also gives the result of execution of other test scripts.

Note: assertAll() should be given mandatorily at the end of each block while using SoftAssert.

Differences between hard Assert and SoftAssert

Assert	SoftAssert
1. It contains all static methods	1. It contains all non-static methods
2. When AssertionError occurs it skips the execution of remaining statements in the block	2. When AssertionError occurs it still executes the remaining statements of the current block
3. assertAll() method doesnot exist	3. assertAll() method is mandatory and should be given at the end of the block

Chapter -4

TestCase 1:
Open the browser
Enter skillrary.com
Click on 'GEARS' tab
Click on SKILLRARY DEMO APP
MouseHover to course tab
Select 'Selenium training'
Double click on '+' button
Click on Add to cart
Handle alert popup
close the browser

Testcase 2:
Open the browser
Enter skillrary.com
Click on GEARS tab
Click on SKILLRARY DEMO APP
Select 'Testing' from category dropdown
Drag and Drop 'JUnit' course to 'MyCart'
Scroll the page till facebook icon and
click facebook icon
close the browser

TestCase 3:
Open the browser
Enter skillrary.com
Type 'core java for selenium' in search tab
Click on search button
Click on 'Core java for selenium' course
Click on play button
Click on pause button
Click on 'Add to wishlist'
Close the browser

TestCase 4:
Open the browser
Enter skillrary.com
Click on GEARS tab
Click on SKILLRARY DEMO APP'
Scroll till the end of the page
Click on contact us
Enter all the details
Click on 'send us mail'
Close the browser

Chapter - 4

Framework/Project

Framework is the collection of reusable methods and element repositories and respective business libraries in well organized manner which facilitates faster test script development and easier test execution and generates reports automatically.

Maven Project:

Maven is build management tool.
We have other build management tools in market - Ant, Gradle
Maven is widely used in real time.

Common Actions:

1. child browser pop up
2. Mouse hover
3. Double click
4. Alert popup
5. close
6. launch browser
7. dropdown
8. drag and drop
9. scroll page
10. frames
11. excel actions
12. Properties file actions

Advantages of Maven project:

1. Handles dependency jar files and plugins.
   pom.xml (project object model xml file) is the heart of maven project where we can add all the dependencies and plugins.
2. Provides well organized folder structure.
3. It supports Jenkins.
4.

Maven has two types of plugins

1. Maven compiler plugin: It is used to compile the project. It is present by default in eclipse. We need not add it explicitly.
2. Surefire plugin: It is used for execution of xml files. It should be added explicitly to the pom.xml file.

Maven folder structure:

1. src/main/java - We store generic libraries and POM classes here.
   Generic libraries - classes and methods for all common and reusable actions.
   POM classes - We store page wise elements and their business libraries.
2. src/main/resources - During execution downloaded jar files and driver executable files are stored here temporarily. Documents related to framework is also stored here.
3. src/test/java - We store all the test scripts here
4. src/test/resources - We store the resources to read data from external files like Properties file and Excel

Framework has 3 stages

1. Design: Here we design the framework architecture.
   - Add all the dependencies and plugins in pom.xml
   - Add the test resources
     1. Properties file
     2. Excel file
   - Develop generic libraries. Create classes for
     1. PropertiesFileUtility
     2. ExcelFileUtility
     3. WebDriverUtility
     4. IConstantPath interface
2. Development: We create POM classes page wise in the application and develop the test scripts using generic libraries and POM classes.
3. Execution: Execute the test scripts in batch as well as in Jenkins

```
@BeforeSuite: We should specify all the before suite configurations like
              establishing database connectivity and other system
              configuration.
@BeforeTest: It is used for parallel executions
@BeforeClass: We should initialize all the generic library classes and
              perform operations like launching browser.
@BeforeMethod: We should initialize all POM pages and navigate to the
               application
@Test: Actual test script
@AfterMethod:
@AfterClass: close browser and close excel
@AfterTest: It is used for parallel executions
@AfterSuite: close the database
```