INTRODUCTION

What is software testing?

Software testing is a process of finding the bugs/defects in the software. The main objective of software testing is to deliver the quality product to the customer. The goal is to release quality software to the customer/client.

Software testing is done in two ways

1. Manual Testing
2. Automation Testing
3. Manual Testing: In this we have to manually test all the test cases and find out the defects. Performing the testing without using any tools is known as Manual Testing. Here tester/person will do the testing.
4. Automation testing: In this we use tools to test the test cases and find defects in the project.

Why should we go for automation testing instead of manual testing?

While performing manual testing we come across a lot of challenges. To overcome this challenges we use automation testing. The main challenge that occurs is in retesting(Doing testing of the test case again and again i.e repetitive activity on the update version of the project done by the developer ) and regression testing(Regression testing is a software testing process conducted after code changes to ensure new updates don’t negatively impact existing functionalities).

What are major challenges faced in re-testing and regression testing?

In re-testing, repetition is involved, so same test case is performed multiple times on the updated project, and in regression testing we have to check where the updated modiule is not affecting the othe modules functionalities. In both the cases the major problem that we face is that we have to put a lot of time and we have to put lot of efforts. To overcome these 2 challenges automation tools were introduced.

Initially automation tools were used to replace re-testing and regression testing. But now for testing automation tools are preferred for all testing types.

What is automation testing?

Performing testing by taking help of automation tool . Automation tools are the tools that are used for testing, the tools are given instructions to be performed.

How would automation tool overcome the challenge of manual testing i.e time and effort?

Once you automate the test case , you can run the test cases n number of times without putting your time and effort. So wherever you feel the test cases are repeated, that test cases can be automated.

How automation tool work? And Why programming is required in automation?

In automation testing, we have manual test cases which are converted into a scripting format. So as a automation tester you have to put as instruction to the tool. The tool will understand the instruction and will perform the same action on the application under the test. These instructions are written with the help of programming language . Whichever scripting language the tool understands we have to write the test cases according to that.

Eg: the manual test cases are written in English, but the tool can’t understand the English language , but the tool can understand the Java language . So we have to write all the test cases in java language

What is selenium?

Selenium is a web based automation tool/library. It was originally developed by Jason Huggins in 2004 ThoughtWorks Company. It is an open source application and free to use.

Selenium is collection of multiple components(IDE, WebDriver, Grid)

Selenium components are:-

Selenium web driver: it is used for creating and running automation test scripts

Selenium ide : it is a plugin used for the browser

Selenium grid: it is used to execute the test cases in remote environment

Advantages of selenium

-Open source tool and free

- Multiple operating systems

-Supports multiple browsers

- Support multiple languages (java, python,c#, ruby, js, etc)

-integrate third party tools in to selenium

Selenium is used for testing the test cases, it will not generate an report. So we use testing, which is a third party tool.

If selenium doesn’t have any features , it has the capability to integrate with third party tool

Disadvantages/limitations of selenium

-Cannot support windows based applications (Autoit, Sikuli, Robot api)

- Reporting not supported(TestNG, Extent reports)

-Cannot support excel files(Apache Poi)

-graphs, captua

**LOCATORS**

//check the img in video

A diagram of a computer

Description automatically generated

1) Identifying elements - Locators

2) Perform actions - Methods

Locator - Locate the web elements

findElement()

findElements()

length - arrays a.length

lenght() - string s.lenght()

size()- List, Set, HashMap

Difference between findElement(loc) and findElements(loc):-

findElement(loc) ----> WebElement

findElements(loc) ----> List<WebElement>

Scenario1: locator is matching with single web element.

-------

findElement(loc) ---->single webelement WebElement

findElements(loc)--->single webelement List<WebElement>

Scenario2: locator is matching with multiple web elements

------------------

findElement(loc)---->single webelement(first) WebElement

findElements(loc) ---> multiple web elements List<WebElement>

Scenario3: locator is incorect ( No elements are not matching with locator)

------

findElement(loc)----> NoSuchElementException

findElements(loc)--->returns zero

**CSS Selector : Eclipse**

What is XPath in Selenium?

XPath in Selenium is an XML path used for navigation through the HTML structure of the page. It is a syntax or language for finding any element on a web page using XML path expression. XPath can be used for both HTML and XML documents to find the location of any element on a webpage using HTML DOM structure.

Xpath is generated at runtime i.e when the webste is running the webpage is generated

In Selenium automation, if the elements are not found by the general locators like id, class, name, etc. then XPath is used to find an element on the web page.

There are two types of XPath:

1. Absolute XPath(full path): It is the direct way to find the element, but the disadvantage of the absolute XPath is that if there are any changes made in the path of the element then that XPath gets failed.

In inspect->copy full xpath

1. Relative XPath(partial path): It can search elements anywhere on the webpage, means no need to write a long xpath and you can start from the middle of HTML DOM structure. Relative Xpath is always preferred as it is not a complete path from the root element. Relative xpath will use attributes for defining the xpath

In inspect->copy xpath

Which one will you prefer?

We will prefer relative xpath because in absolute xpath every element is representing some element in the DOM , we need the traverse each and every element in the xpath . suppose a developer removes /add an element in the webage then the full absolute xpath will be of no use

How to write your own xpath?

Syntax: //tagname[@attribute=’value’] or //\*[@attribute=’value’] *We use \* when we don’t know the tag name*

Eg: //input[@type="text"]

If the element doesn’t have any attribute then we will use the parent’s attribute and then go to the desired tag

Eg: //form[@method="get"]/input

How to generate xpath automatically?

1. Using developer tool (i.e inspect)
2. Using selector-hub extension

*When the elements have the same attribute name then it is better to use the text element of the tag*

*Eg: <a href="/apple-macbook-pro-13-inch">Apple MacBook Pro 13-inch</a>*

*Here “apple macbook pro 13” is the text element*

*To use this in xpath we use*

*//a[text()="Apple MacBook Pro 13-inch"]*

*Text() is one of the methods of xpath functions*

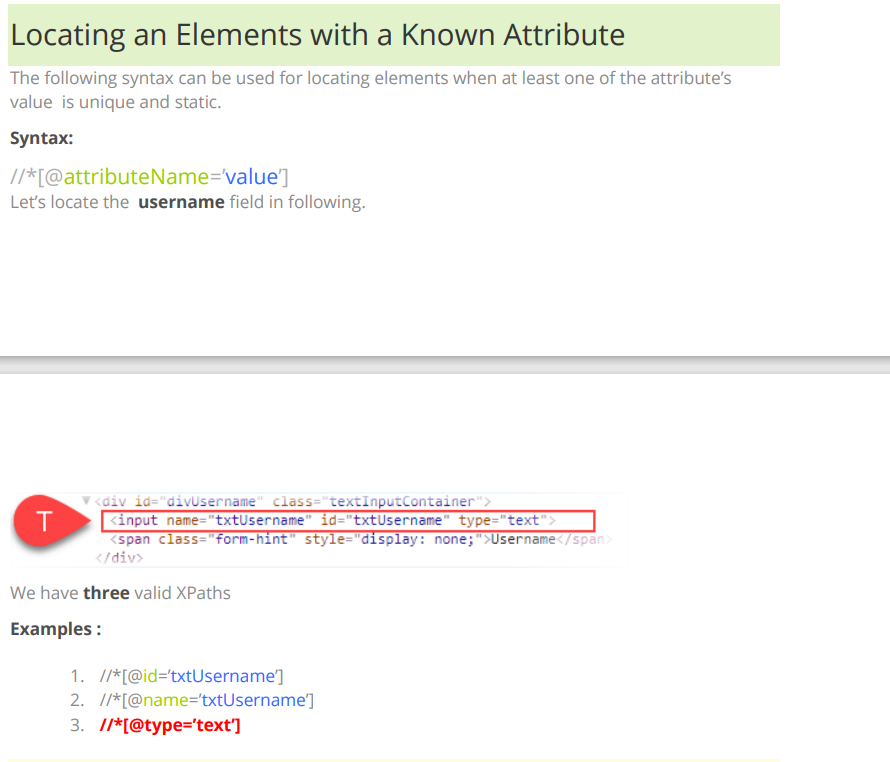
*Similarly we have normalize-pace(), contains(), start-with() some other functions of xpath*

*Normalize-pace() ignores the spaces in the text where as text() doesn’t ignore the spaces*

*Xpath operators: and or*

*Xpath axes – following, preceding, sibling, parent, self, ancestor etc*

**LECTURE 22**

****

How does Selenium works when there are multiple elements (candidates) for an XPath?

Selenium will pick the first element in the path if there are multiple candidates for a given XPath when webdriver.findElement(By.xpath("XPATH")) method is used. All the candidate elements can be assigned to a List when webdriver.findElements(By.xpath("XPATH")) method is used.

1. To select the third input element: //form/input[3]

2. To select the last input element: //form/input[last()]

Locating elements by position is discussed further in a separate section.

findElement method throws NoSuch ElementException error when there is no elements with given XPath. Use findElements method and check the size when working with non-present elements.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

*Refer Xpath.pdf*

*Video for code*

**LECTURE 23: Web driver methods**

1. Get methods : We use “get methods” through web driver instance
2. Conditional methods: The commands are accessed through web elements and not from driver.
3. Navigational methods: [Selenium WebDriver - Navigation Commands - GeeksforGeeks](https://www.geeksforgeeks.org/selenium-webdriver-navigation-commands/)
4. Browser methods
5. Wait methods
6. Get Methods
   1. get(url) : It is used to open the link in the web driver browser
   2. getTitle(): Gets the title of the current web page.
   3. getCurrentUrl(): Gets a string representing the current URL that the browser is opened.
   4. getPageSource(): Get the source of the currently loaded page. We use to to validate whether an element is present in the html page or not. Whenever you are writing an xpath we have to make sure that we write it according to DOM
   5. getWindowHandle(): It is used to find A Target Locator which can be used to switch or select a frame or window
   6. getWindowHandles(): It returns A set of window handles which can be used to iterate over all the open windows. Iset is the datatype of the windowhandles().
7. Conditional methods: These commands will return either true/ false
   1. isDisplayed(): This method will return true , when the elements are present on the web page
   2. isEnabled(): sometimes the elements are enabled in the web page , sometimes the elements are disabled eg: checkboxes. So if the element is enabled then the it will return true else it will return false
   3. isSelected(): this method will return true if the web element is selected eg radio button
8. Navigational methods
9. Browser methods: There are two methods i. close() closes a single browser and quit() closes all windows in the browser
10. Wait methods

**LECTURE 24: Wait commands**

Wait statements will be used to solve synchronization problem. *synchronization refers to coordinating the timing of script execution with the application. It ensures that the code and applications work together seamlessly to carry out desired operations.*

There are 2 types of wait commands

1. Implicit wait
2. Explicit wait/Fluent state *fluent state and explicit wait are similar but fluent state has more features than explicit wait*

*Thread.sleep(milliseconds) it is a function of java , it doesn’t come under wait commands*

When we have Thread.sleep() method then why do we need wait commands?

The advantages of sleep command is that it is easy to use.

But the disadvantage is that when the website is loaded in 2 seconds but the sleep timer is of 5seconds then the website has to wait for 5seconds which will cause performance issue. And suppose the sleep timer is not sufficient then you will get an exception.

* Implicit wait: implicit Wait directs the Selenium WebDriver to wait for a certain measure of time before throwing an exception.

We use this command the following command

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

implicit wait is applicable for every element in the automation script i.e if the element is not available it will wait for 10seconds and then throw an NoSuchElementFound exception

If the element is available it will move to the next line

Duration can be in seconds, minutes, hours

eg:

WebDriver driver = new ChromeDriver();

driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/auth/login");

//Thread.sleep(5000);

//implicitly wait

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

driver.findElement(By.xpath("//input[@placeholder='Username']")).sendKeys("Admin");

driver.findElement(By.xpath("//input[@placeholder='Password']")).sendKeys("admin123");

*Here in this example, the wait time is of 10seconds. . If they don’t find the username element then after 10seconds the driver will throw an error message, but if they found the username element in 2 second then the next line will be executed.*

*When we use Thread.sleep() we have to declare it before every element*

Advantage of implicit wait

* We have declare the implicit wait only for a single time
* It will not wait for maximum time if the element is available
* Applicable for all the elements
* Easy to use

Disadvantage

* If the time is not sufficient then you will get an exception

Explicit wait

[Selenium Wait – Implicit, Explicit and Fluent Waits (guru99.com)](https://www.guru99.com/implicit-explicit-waits-selenium.html)

**LECTURE 25**

* Checkboxes
* Dropdowns
* Checkboxes: Refer video lecture
* Dropdown

Dropdown in a website are of 3 types

* 1. **Drop down having select tag in DOM**
  2. **DropDown not having select tag in DOM(input/div) – Bootstrap dropdown**
  3. **Auto Suggest drop down(Dynamic)**
  + **Drop down having select tag in DOM**

//how we will select options from the drop down

We will use the

// we have selected the select tag and saved it the variable

WebElement drpCountryEle = driver.findElement(By.*xpath*("//select[@id='input-country']"));

Select drpCountry = **new** Select(drpCountryEle);

* “Select” is a class provided by Selenium WebDriver. It is used to handle dropdown menus on web pages.
* “drpCountry” is an object of the Select class. This object will be used to interact with a specific dropdown menu on a web page.
* “new Select(drpCountryEle)” is creating a new instance of the Select class. The Select constructor takes a WebElement as an argument.
* “drpCountryEle” is expected to be a WebElement that represents the dropdown menu on the web page. This would typically be found using one of the findElement methods from the WebDriver class, like driver.findElement(By.id("id")) or driver.findElement(By.name("name")).
* *WebElement drpCountryEle = driver.findElement(By.xpath("//select[@id='country-list']"));*

*Select drpCountry = new Select(drpCountryEle);*

* *Explain me this code*
* *Sure, I’d be happy to explain this code. This is a piece of code written in Java for Selenium WebDriver, a popular tool for automating browsers.*
* *Here’s what each line does:*
* *WebElement drpCountryEle = driver.findElement(By.xpath("//select[@id='country-list']")); This line is finding a web element on the page using its XPath. The XPath //select[@id='country-list'] is looking for a <select> HTML element with the id country-list. This element is then stored in the drpCountryEle variable.*
* *Select drpCountry = new Select(drpCountryEle); This line is creating a new instance of the Select class, which is a special class in Selenium that represents a dropdown menu. It’s being created with the drpCountryEle WebElement, which means this Select object is tied to the dropdown menu on the webpage.*
* *We use select class only when the the html code has select tag*
* *i.e:*

*A white screen with red and blue text

Description automatically generated*

Operations of select

* Selecting an option from the select dropdown

There are 3 methods of selecting an option from the select tag

* + selectByVisibleText(): Selects/deselects the option that displays the text matching the parameter.

Parameter: The exactly displayed text of a particular option

Eg:

Select drpCountry = new Select(driver.findElement(By.name("country")))

drpCountry.selectByVisibleText("ANTARCTICA");

* + selectByValue() and deselectByValue(): Selects/deselects the option whose “value” attribute matches the specified parameter.

Remember that not all drop-down options have the same text and “value”, like in the example below.

Parameter: value of the “value” attribute

Eg:

A group of words with arrows

Description automatically generated with medium confidence

drpCountry.selectByValue("234");

* + selectByIndex() and deselectByIndex(): Selects/deselects the option at the given index.

Parameter: the index of the option to be selected.

Eg: drpCountry.selectByIndex(0);

Refer lecture25\_dropdown/HandleDropDownWithSelectTag.java

* + **DropDown not having select tag in DOM(input/div) – Bootstrap dropdown**
    - Refer lecture25\_dropdown/HandleDropDownWithoutSelectTag.java
    - Refer lecture25\_dropdown/Amazon.java
  + **Auto Suggest drop down(Dynamic): next lecture**

**LECTURE 26:**

AUTO SUGGEST DROP DOWN(DYNAMIC)

Here the options that you get is not constant, it may change . Also the options that you are getting now its not necessary that after some time the same options will not occur

Refer:lecture26/AutoSuggestDropDown.java

Driver.switchTo().alert()

ALERT/POPUPS

1. Alert with ok
2. Alert with ok and cancel
3. Alert with inputbox along with ok and cancel
4. Alert with no elements

Refer : Video

**LECTURE 27**

Handles Frames/IFrames and Window Browser

Frames : driver.switchTo().frame()

Refer: [Switching Between Frames Using Selenium WebDriver in Java | Baeldung](https://www.baeldung.com/java-selenium-change-frames)

Refer : Video

IFrames:

Refer: [How to Handle iFrames in Selenium Webdriver: switchTo() (guru99.com)](https://www.guru99.com/handling-iframes-selenium.html)

Refer : Video

Browser Window:

Refer: [How to handle multiple windows in Selenium? - GeeksforGeeks](https://www.geeksforgeeks.org/handle-multiple-windows-in-selenium/)

Close(): close single browser window which is focused on the driver

Quit(): close all the browser windows which are currently opened

**LECTURE 28**

**HANDLES STATIC AND DYNAMIC WEB TABLES**

WEB TABLE : There are 3 types of Table

1. Static Table :
2. Dynamic Table :
3. Pagination Table :

* Static Table :

As the website is not working this is the table

|  |  |  |  |
| --- | --- | --- | --- |
| Book Name | Author | Subject | Price |
| Learn Selenium | Amit | Selenium | 300 |
| Learn Java | Mukesh | Java | 500 |
| Learn JS | Animesh | Javascript | 300 |
| Master in Selenium | Mukesh | Selenium | 3000 |
| Master in Java | Amod | Java | 2000 |
| Master in JS | Amit | Javascript | 1000 |

Refer : lecture28/statictable.java

* Dynamic and Pagnation table:

This is pagnation

A number on a white background

Description automatically generated

When we click on each page the elements/data in the table changes

This changing data in the table is called the dynamic table

Refer: lecture28/DynamicAndPagnation.java

**LECTURE 29**

DATETIME PICKER

There are 2 types of date web elements

* 1. Standard web elements (normally that we get in html)
  2. Customized web elements (created using drop down of month and year)

Current date

Past date(for dob)

Future date(for travel)

Refer: video lecture

HIDDEN DROPDOWN

In this when you click the drop down and the data don’t disappear in inspect then we have to go to

Inspect->event listener->blur option->remove

Refer : video lecture

//Assignment remaining

**LECTURE 30**

MOUSE ACTIONS

Mouse actions are mouse hover,right click, left click , drag and drop,

We will “actions” class. This class is provided by selenium webdriver

Syntax: Actions act = new Actions(driver);

Here we need to pass driver in the action class

1. Mouse hover: moveToElement(Web Element)

Refer java code

1. Right click: contextClick(element)

*Right click is also called as contextclick*

Refer java code

1. Double click: doubleClick(Web element)

*Html code*

A screenshot of a computer

Description automatically generated

getText() method always returns the inner text of the element

eg: <input id=”abc”>testing</input>

testing is the inner text . so getText() will return “testing”

but when the web element is

<input id=”abc” value=”testing” />

When we will use getText() method it will return null . So we need to use

grtAttribute(“value”) to get the text of the we element i.e testing

when we use getAttribute(“id”) “abc” would be returned

Refer java code

1. Drag and drop

Refer java code

1. Slider: here we have to imagine the slider as a line in the coordinate axis

Syntax: act.dragAndDropBy(min\_slider, 0, 0)

Min\_slider: it is the web element where the pointer on the web page is pointing

First 0: it is the x coordinate. This is also used to calculate the distance from the left side. When we take the silder pointer from 0 the x coordinate is positive(in minimum case) and when we take the slider pointer from the end then we take the x coordinate as negative(maximum case)

Last0 : it is the y coordinate

Refer java code

**LECTURE 31**

HANDLE KEYBOARD EVENTS:-

Usually when we are working we use certain keys like ctrl+a, ctrl+c, tab , etc

To handle such keys in website dynamically we use actions class.

Eg: In this we will write a text, select the text using ctrl+a, will copy the text, then press tab, then ctrl+v ,

Refer : java code

JAVASCRIPT EXECUTOR

What is JavaScriptExecutor?

JavaScriptExecutor is an Interface that helps to execute JavaScript through Selenium Webdriver. JavaScriptExecutor provides two methods “executescript” & “executeAsyncScript” to run javascript on the selected window or current page.

WHEN YOU GET elementIntercepted method expection , it means that the element called and the callable method there is a difference ,t0hen we should know that we need to use javascript wxecutor

Why do we need JavaScriptExecutor?

In Selenium Webdriver, locators like XPath, CSS, etc. are used to identify and perform operations on a web page.

In case, these locators do not work you can use JavaScriptExecutor. You can use JavaScriptExecutor to perform an desired operation on a web element.

Also when we use sendkeys method , after the input it should call the js function , but it takes delay to call it, then we get elementIntercepted error

Selenium supports javaScriptExecutor. There is no need for an extra plugin or add-on. You just need to import (org.openqa.selenium.JavascriptExecutor) in the script as to use JavaScriptExecutor.

JavaScriptExecutor Methods in Selenium

executeScript

This method executes JavaScript in the context of the currently selected frame or window in Selenium. The script used in this method runs in the body of an anonymous function (a function without a name). We can also pass complicated arguments to it.

The script can return values. Data types returned are

* Boolean
* Long
* String
* List
* WebElement.

JavascriptExecutor syntax:

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript(Script,Arguments);

Script – This is the JavaScript that needs to execute.

Arguments – It is the arguments to the script. It’s optional.

ExecuteAsyncScript:

With Asynchronous script, your page renders more quickly. Instead of forcing users to wait for a script to download before the page renders. This function will execute an asynchronous piece of JavaScript in the context of the currently selected frame or window in Selenium. The JS so executed is single-threaded with a various callback function which runs synchronously.

Refer: [JavaScriptExecutor in Selenium with Example (guru99.com)](https://www.guru99.com/execute-javascript-selenium-webdriver.html)

Video lecture for code and an img

SCROLLING PAGE:

Refer: [JavaScriptExecutor in Selenium with Example (guru99.com)](https://www.guru99.com/execute-javascript-selenium-webdriver.html)

Video lecture for code

**LECTURE 32**

In this lecture we will cover some important topics like broken links, how to capture screenshots

BROKEN LINKS:

Normally, when you click on the link, one request will go to the server and then the server will give you some response on the web page.

Every link has an href attribute which consist the web page link

What is the difference between broken link and one link?

Refer: [How to Find Broken Links in Selenium (guru99.com)](https://www.guru99.com/find-broken-links-selenium-webdriver.html)

Video lecture for code

CAPTURE SCREENSHOT:

Refer: video lecture

HEADLESS BROWSER

When you run the headless java code , the browser won’t open , but testing the functionalities will be done

Refer: theory: [Headless Browser Selenium (HTMLUnitDriver) (guru99.com)](https://www.guru99.com/selenium-with-htmlunit-driver-phantomjs.html)

Practical: [Headless Browser Testing with Selenium: Tutorial | BrowserStack](https://www.browserstack.com/guide/selenium-headless-browser-testing)

**LECTURE 33**

DATA DRIVEN TESTING

This testing is used for the data present in the excel file

To read and write data in excel sheet we use Apache library called POI .

This library is used for automating ms documents – excel, reading data from excel, writing data in excel

We need to download apache poi using dependency in pom.xml file

To read the data in the excel file we have to follow this steps

Excel File-->Workbook-->Sheets---->Rows--->Cells

Excel File: for reading and writing a file we use

FileInputStream: it is used to read a file

FileOutputStream: it is used to write data in the file

WorkBook : to read the workbook we use “XSSFWorkbook”

Sheets: to read the sheet we use “XSSFSheet”

Rows: to read the rows we use “XSSFRow”

Cell: to read each cell we use “XSSFCell”

**LECTURE 34**

For data driven testing we need to use utility file. It is a java file which consists of all the methods/ functions such as getting no of rows, no of coulmns , last row, writing data, reading data etc i.e reusable methods , this methods are used

So we can call this te

**LECTURE 35**

INTRODUCTION TO TESTNG(Test New Generation)

TestNG is an automation testing framework in which NG stands for “Next Generation”. TestNG is inspired by JUnit which uses the annotations (@). TestNG overcomes the disadvantages of JUnit and is designed to make end-to-end testing easy.

TestNG is a testing framework inspired from JUnit and NUnit but introducing some new functionalities that make it more powerful and easier to use. TestNG uses annotations to control the execution flow of tests.

Using TestNG, you can generate a proper report, and you can easily come to know how many test cases are passed, failed, and skipped. You can execute the failed test cases separately.

For example:

Suppose, you have five test cases, one method is written for each test case (Assume that the program is written using the main method without using testNG). When you run this program first, three methods are executed successfully, and the fourth method is failed. Then correct the errors present in the fourth method, now you want to run only fourth method because first three methods are anyway executed successfully. This is not possible without using TestNG.

The TestNG in Selenium provides an option, i.e., testng-failed.xml file in test-output folder. If you want to run only failed test cases means you run this XML file. It will execute only failed test cases.

While we are running any automation tests using Selenium or any other tool, we will be required to view and analyze the results of the execution to conclude the number of tests which got executed, passed, failed, failure data, and so on in the form of reports.

Sometimes, the screenshots where the tests have undergone a failure are also captured in the reports. The test reports are also required to be shared with the project stakeholders on a periodic basis. For this purpose, we can take the help of the TestNG reports.

A TestNG report is a html report generated automatically once a test case built and run with the help of the TestNG. It is a unit testing framework which can be integrated with Selenium tests and used for reporting purposes.

In addition to this, the TestNG has a default reporting class called the Reporter which helps to log. This is useful in detecting the root cause of failure to debug the failed test.

TestNg is a predefined framework which provides n number of features that are useful for automation. It is a java based unit testing tool

Before TestNg we used to use Junit

Developers use TestNg for unit testing

Why Use TestNG with Selenium?

Default Selenium tests do not generate a proper format for the test results. Using TestNG in Selenium, we can generate test results.

Most Selenium users use this more than Junit because of its advantages. There are so many features of TestNG, but we will only focus on the most important ones that we can use in Selenium. Following are the key features of Selenium TestNG:

* Generate the report in a proper format including a number of test cases runs, the number of test cases passed, the number of test cases failed, and the number of test cases skipped.
* Multiple test cases can be grouped more easily by converting them into testng.xml file. In which you can make priorities which test case should be executed first.
* The same test case can be executed multiple times without loops just by using keyword called ‘invocation count.’
* Using testng, you can execute multiple test cases on multiple browsers, i.e., cross browser testing.
* The TestNG framework can be easily integrated with tools like TestNG Maven, Jenkins, etc.
* Annotations used in the testing are very easy to understand ex: @BeforeMethod, @AfterMethod, @BeforeTest, @AfterTest
* WebDriver has no native mechanism for generating reports. TestNG can generate the report in a readable format like the one shown below.

ADVANTAGES OF TEST NG

* + 1. Test Cases and test suites:

Foe every test case we are creating a class and doing testing on it and checking the result. These test cases are a part of test suite. Test suite is a collection of test cases to run a single job with different test scenarios. In real time there are many test cases and each test case should be part of test suites, we can divide this test cases into multiple groups . So whenever we can to execute specific test cases we should be able to execute , for that we need to maintain the test cases and test suite. This maintenance can be easily done by testng .

* + 1. Grouping of test cases:

We can group the test cases into multiple suites or groups

* + 1. Prioritize:

After grouping we need to prioritize the test cases i.e which test case should prioritize first

* + 1. Parameterization:

When we are executing test cases we need to send parameters to check the test case

Eg: at runtime we can pass which browser should the website be runned

* + 1. Parallel Testing:

In parallel testing we can test a test case in multiple browser at a same time. Normally it is not possible so we have run the test case on different browser one after the other

* + 1. Reports:

Testng helps us to make reports

HOW TO SETUP TESTNG?

1. Help menu-> Eclipse marketplace 🡪 search for testng for eclipse 🡪 install

Or

Window🡪preferences🡪testng

1. Add testng dependency to pom.xml
2. Pacakage 🡪 classes 🡪 TestMethods

**WE DON’T HAVE MAIN METHOD IN TESTNG**

1. TestSuite 🡪 Test Classes🡪Test Steps //TestSuite is nothing else but like a package
2. Package🡪 classes🡪TestMethods // We will include the test method in the test steps

*@Test is an annotation that is used to declare that the following method is used for testing. If you don’t add @Test annotation then TestNg will not execute the method*

*Java code : lecture 35/FirstTest.java package*

*In XML file , we can test multiple class. Class is nothing else but the java classes*

*The structure of the xml file is*

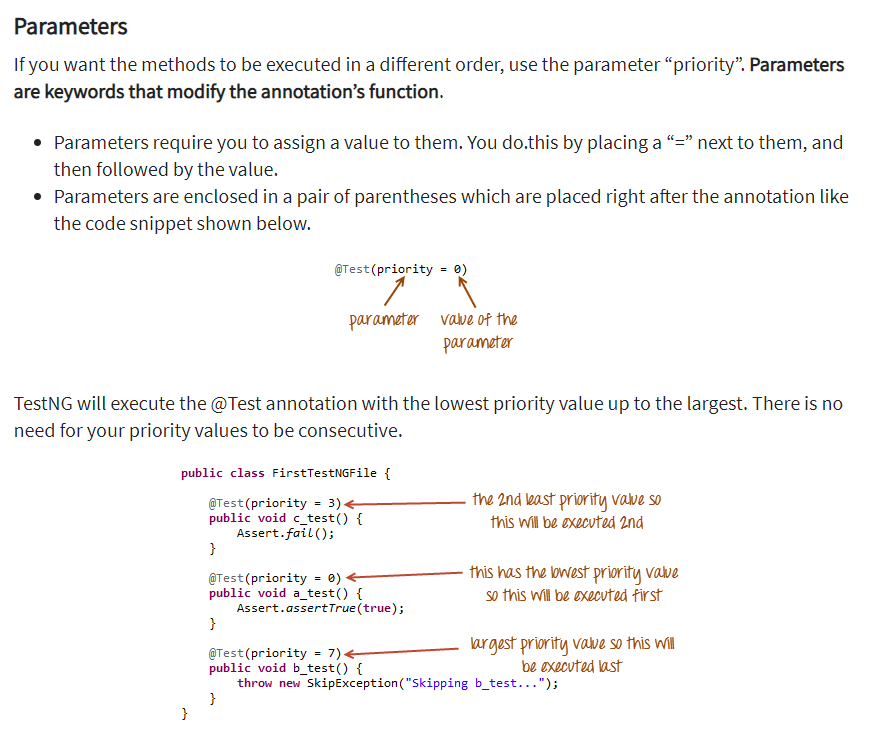
*Suite tag->Test tag -> Classes tag->Class tag*

*Inside the class tag we have to mention the java class name which file needs to be tested*

*When we run the XML file it will generate emailable-report.html(this report is not that attractive, hence no one uses it) and index.html. To access it we have to refresh the project after running the testng.xml file*

*XML is used for testing multiple test cases*

*Priority : control the order of execution*

**

*A cartoon character with a speech bubble

Description automatically generated*

*Java code : lecture 35/SecondTest.java package*

**LECTURE 36**

The hierarchy of testng is , a suite is created, each suite has multiple tests., each test has classes and each class has methods

Suite(XML File)🡪Tests🡪classes🡪Methods

ANNOTATIONS

The different types of Annotations are

A diagram of a diagram

Description automatically generated

* @BeforeSuite: The @BeforeSuite annotated method will run before the execution of all the test methods in the suite.
* @AfterSuite: The @AfterSuite annotated method will run after the execution of all the test methods in the suite.
* @BeforeTest: The @BeforeTest annotated method will be executed before the execution of all the test methods of available classes belonging to that folder.
* @AfterTest: The @BeforeTest annotated method will be executed before the execution of all the test methods of available classes belonging to that folder
* **@BeforeClass:** The @BeforeClass annotated method will be executed before the first method of the current class is invoked.
* **@AfterClass:** The @AfterClass annotated method will be invoked after the execution of all the test methods of the current class.
* **@BeforeMethod:** The @BeforeMethod annotated method will be executed before each test method will run.
* **@AfterMethod:** The @AfterMethod annotated method will run after the execution of each test method.
* Refer : lecture36/AllAnnotations.java

Suppose there is a test case

1. Login
2. Search
3. Logout
4. Login
5. Advance search
6. Logout

As you can see here, the 2 main and different methods are Search and Advanced search. Login and logout methods are repeated plus they are the prerequisites for search and advanced search methods. So we can use @BeforeMethod for login and @AfterMethod for logout, so that we don’t have to write the login and logout methods multiple times.

It will execute login and logout for every method present

So the code structure would be

1) Login - @BeforeMthod

2) Logout- @AfterMethod

3)Search- @Test

4) Advance search-@Test

We don’t need to test the login and logout methods, because they are the entry and exit methods

If suppose you want to test login or logout then you have create a separate test method for checking it

Refer : lecture36/BeforeAndAfterMethods.java

* @Test: It’s used to mark a method as a test method.

BENEFITS OF USING TESTNG ANNOTATIONS:

* TestNG Annotations made the life of testers very easy. Based on your requirements, you can access the test methods, i.e., it has no predefined pattern or format.
* You can pass the additional parameters to TestNG annotations.
* In the case of TestNG annotations, you do not need to extend any test classes.

*(In many older testing frameworks like JUnit, you often have to extend a specific test class (like TestCase in JUnit) in order to gain access to testing methods and lifecycle callbacks. This can lead to limitations because Java does not support multiple inheritances. If your class extends the test class, it cannot extend any other class.)*

* TestNG Annotations are strongly typed, i.e., errors are detected at the compile time.

ASSERTION

Assertion is used for validation/checking. Assertions (also known as Asserts).Usually we use if-else condition to check validation but when we use if-else in test method , even if the condition is failed the test case is passed

Eg:

A screenshot of a computer program

Description automatically generated

Output

A screenshot of a computer

Description automatically generated

In Selenium, assertions are used for verification or checkpoints in the test case. If assertion is not used in the test case, it’s not possible to determine whether the test case is passed or failed. Assertion will be used to generate the test execution reports. And if all the test steps present in the test cases passed successfully, then assertion will not impact the test. It will report only when a test case is failed.

**Difference between Assert and Verify in Selenium**

* **Assert**: Think of it like a strict teacher who stops the class as soon as a student makes a mistake. In testing, if a condition checked by an “Assert” command is not met (i.e., the test fails), the test stops right there and doesn’t continue to the next step.
* **Verify**: This is like a lenient teacher who notes down all the mistakes and tells you at the end of the class. Even if a condition checked by a “Verify” command is not met, the test doesn’t stop. It continues to check the rest of the conditions, and at the end, it will report all the failures that occurred during the test.

So, in simple terms, “Assert” stops the test as soon as a failure occurs, while “Verify” continues the test despite failures and reports all the failures at the end. The choice between using “Assert” or “Verify” depends on what you need for your test. If a failure in a condition means the rest of the test can’t be carried out, use “Assert”. If the rest of the test can still be carried out despite failures, use “Verify”.

Types of assertion

1. Hard

access though 'Assert' class

all methods are static

if hard assertion got failed then rest of the statement will not be executed.

1. Soft

access through 'SoftAssert' object

SoftAssert sa=new SoftAssert();

sa.assertTrue()

if soft assertion got failed then rest of the statemetns still execute.

Using these types we will use the below methods .

Methods of Assertions:

* AssertEquals
* AssertNotEquals
* AssertTrue
* AssertFalse
* AssertNull
* AssertNotNull

[Asser t and Verify Methods in Selenium | BrowserStack](https://www.browserstack.com/guide/verify-and-assert-in-selenium)

Hard assert and soft assert codes

DEPENDENCY METHODS

While executing a code, if a method is dependent on other method , such methods are called as dependency methods.

Eg: If there is dashboard page which will get open only after the login page, then we can say that the dashboard page is dependent on login page

Syntax: *in the @Test we have to add dependsOnMethods{method\_name}*

Refer: lecture36/DependentMethods.java

The Assert.assertTrue(condition); statement is used in unit testing frameworks like JUnit or TestNG in Java.

Here’s what it does:

It checks whether the condition inside the parentheses is true.

If the condition is true, the assertion passes and the program continues.

If the condition is false, the assertion fails. The program throws an AssertionError and halts the execution of the current test.

This is typically used to verify the correctness of a condition in a unit test. If the condition is not met, the test fails, indicating that there may be a bug in the code being tested.

**LECTURE 37**

GROUPING

TestNG allow you to perform groupings of different test methods. Grouping of test methods is required when you want to access the test methods of different classes.

Not only you can declare the methods within a specified group, you can also declare another group within a specified group. Thus, TestNG can be asked to include a certain set of groups while excluding another set of groups.

It provides you maximum flexibility by partitioning your test methods in groups and does not require recompilation of test cases if you run your two different sets of test cases back to back.

Groups are specified in the testng.xml file with <groups> tag. Groups can be specified either in the <suite> tag or <test> tag.

When executing grouping we have to run only the xml file

Refer: lecture37/grouping.java, testng.xml

PARAMETERIZATION

Parameterization is used to pass parameters to the test

2 ways to execute parameterization

1. Data Provider
2. XML file

1) Data provider

[How to use TestNg DataProviders (with Detailed Examples) in Selenium (toolsqa.com)](https://www.toolsqa.com/testng/testng-dataproviders/)

Refer java code: lecture37/DataProviderProject

**LECTURE 43**

**Invocation count**

In TestNG, the @Test(invocationCount=x) annotation allows you to define the number of times a test method should be executed. The x value specifies how often the test method should be called. If you want to run the same test multiple times, you can use the invocationCount attribute

We use this for exhausting testing. Exhaustive testing is nothing else but , sometimes your application’s functionality may not be stable , sometimes it will work sometimes it will not work, so to identify such type of scenario we will execute the same type of code multiple times to find where the error is exactly occurring . We can even execute the code more than 100 times.

Eg:

A screen shot of a computer

Description automatically generated

Output

A screenshot of a computer

Description automatically generated

**Creating Parameterization using xml**

Using XML we can achieve parallel testing.

When we want to execute more no of test cases in less time we prefer using parallel testing

Serial execution:

Parallel execution: [TestNG Parallel Execution - How to run Selenium tests in parallel? (toolsqa.com)](https://www.toolsqa.com/testng/testng-parallel-execution/)

Refer video lecture for code

STEPS:

1. Create a test case method
2. Create an xml file to run the test case
3. Pass browser name parameter from the xml to setup() method
4. Execute test case on chrome and edge (serial execution)
5. Execute test case on chrome and edge (parallel execution)

Refer: video lecture

**LECTURE 44**

TestNG listeners

What is testNg listeners?

TestNG listeners are the piece of code that listens to the events occurring in the TestNG. If the event matches the event for which we want the listener to listen, it executes the code, which ultimately results in modifying the default behavior of TestNG. For example, we want to print the exception error onto the reports only if the test fails. Here, we can apply a TestNG listener that will listen to the event of "failing of test case" and when it does, it will log the error.

When you execute the test cases, we get result like the test case is pass,failed or skipped . After getting this result we want to perform some actions, these actions are called as post actions, generating a report is a post action,and to define these post actions we use testng listeners

A diagram of a computer

Description automatically generated

For using TestNg listeners we use “ITestListener” interface

TestNG provides a bunch of listeners as a part of its testing environment. These listeners are as follows:

1. *ITestListener*
2. *IReporter*
3. *ISuiteListener*
4. *IInvokedMethod*
5. *IHookable*
6. *IConfigurationListener*
7. *IConfigurableListener*
8. *IAnnotationTransformer*
9. *IExecution*
10. *IMethodInterceptor*

**ITestListener In TestNG**

ITestListener is the most used listener in TestNG with Selenium webdriver. The ITestListener implements since it is an interface, and the class in which we implement the listener overrides the ITestListener methods. ITestListener listens to specific events (depending on its methods) and executes the code written inside the method. With ITestListener in TestNG, we can also log the events onto the reports using the Selenium web driver.

TestListenerAdapter--🡪 class and the ITestListerner interface consists the following methods

Imp methods are

* onTestStart()
* onTestFailure()
* onTestSuccess()
* onTestSkipped()

//These method are abstract methods in the interface and in the class

So we need to override this methods

So we need to create our class and implement the interface or extend the TestListernerAdapter class, either of them can be used

But it is preferred to use the interface

The *ITestListener* contains the following methods:

* ***onStart***: *This method invokes when the test class is instantiated and before executing any test method*.

***Syntax***: ***void onStart(ITestContext context);***

* ***onFinish***: *This method invokes when all the test methods have run, and calling of all of their configuration methods happens*.

***Syntax***: ***void onFinish(ITestContext context);***

* ***onTestStart***:  *This method invokes every time a test method is called and executed*.

***Syntax***: ***void onTestStart(ITestResult result);***

* ***onTestSuccess***: *This method is invoked every time a test case passes (succeeds)*.

***Syntax***: ***void onTestSuccess(ITestResult result);***

* ***onTestFailure***: *This method invokes every time a test case fails*.

***Syntax***: ***void onTestFailure(ITestResult result);***

* ***onTestSkipped***: *This method invokes every time a test skips*.

***Syntax***: ***void onTestSkipped (ITestResult result);***

* ***onTestFailedButWithinSuccessPercentage***: *This method invokes when the test method fails as a whole but has passed a certain success percentage, which is defined by the user*.

***Syntax***: ***void onTestFailedButSuccessPercentage (ITestResult result);***

Now in the above syntaxes, you must be wondering about the words ***ITestContext*** and ***ITestResult***. So, the term ***'ITestResult' is an interface that describes the result of the test***. Therefore 'result' has been passed as its instance in the syntax. Whereas ' ITestContext ' ***is a class that defines an instance*** 'context', ***which contains all the information about xthe test run***. We can use this information to pass to our listeners, and they can proceed with their queries.

Extent Report

Extent report is used for making reports.

Extent doesn’t beong to selenium or testng

It is a third party report

For all test cases only one report is been generated

When will you generate the report? Before starting the test case or after starting the test case?

Not before not after, before starting the test case the UI of the report will be generated. Once the test case has started executing, depending upon the result(i.e pass, failed, skip) we will update the result in the report.

//code

Without the listener tag in xml file we can implement the listener class in the java class by adding the @Listeners(day38.ExtentReportManager.class) at the initialization of the class

But the disadvantage is that if we have multiple test cases class then we have write the @ at the start of the every class and if I change the listener class name then I have to change it for all the classes

So it is prefreed to used listener in xml file

**LECTURE 48 : POM**

POM is nothing else but a pattern of organizing the web elements. So without duplication, without repetition once you identify the element use those elements in multiple test cases not only for single test case

In POM we will properly maintain the page objects

What problems do we face if we don’t create a page object model?

//video

Page class consists of constructors, locators and action methods

Test class consist of test methods and validation i.re assertion

BROWSER

PAGE CLASS

TEST CLASS

The test class will invoke the page class and the page class wil interact with browser

For every page class we have to crate an page object

2 appraoches to create page object classes

-----------------------------

* 1. Without using Page Factory(Normal appraoch)

//code login page and logintest1

* 1. Using Page Factory
* //code login page and logintest2

PageFactory.initElements(driver, this); is used in the concept of **Page Object Model (POM)** in Selenium. Here’s what it does:

* PageFactory: It’s a class provided by Selenium WebDriver, used to initialize the elements of the Page Object or instantiate the Page Objects itself.
* initElements: This method is used to initialize the elements of the page object.
* driver: This is the WebDriver reference that drives the browser.
* this: It refers to the current class object. In the context of POM, this would refer to the current page object.