Module Name: Selenium WebDriver

Trainer Name: Ankush Vankore

Total Sessions: 19

* Selenium WebDriver
  + Automation Concepts
  + Configuration
  + Launching Different Browsers
    - Chrome
    - Edge
    - Firefox
    - Safari
  + Basic Methods of WebDriver
    - get()
    - getTitle()
    - getCurrentUrl()
    - getPageSource()
    - close()
    - quit()
  + Locators
    - Name
    - Id
    - ClassName
    - CssSelector
    - XPath
    - LinkText
    - PartialLinkText
    - TagName
  + Handling different controls
    - Text box
    - Command button
    - Check box
    - Radio button
    - Hyperlinks
    - Drop down list
    - List box
  + Synchronization
    - Thread.sleep()
    - ImplicitWait
    - ExplicitWait
    - FluentWait
    - PageLoadTimeout
  + Handling tables
  + Handling multiple windows
  + Handling alerts
  + JavascriptExecutor interface
    - Scrolling the page
    - Clicking on controls
    - Entering value in text boxes
  + Handling FileUpload control
  + Robot Class
  + Handling Menus
  + Mouse Actions
    - Click
    - Right click
    - Double click
    - Drag and drop
    - Hovering the mouse
  + Handling iframe
  + Screenshot
* TestNG
  + Advantages
  + Configuration
  + Executing Single test
  + Executing Multiple tests
  + Setting the priority
  + Annotations
    - @Test
    - @BeforeTest
    - @AfterTest
    - @BeforeMethod
    - @AfterMethod
  + Data driven testing via @DataProvider
  + Assertions
  + Reports
    - Normal Report
    - HTML Report
  + Executing test / tests via xml file
  + Parameterization
  + Keyword Driven Framework
  + Modular Framework
  + Page Object Module
  + Data Driven Framework
    - Reading the data from Excel
    - Writing the data to Excel
  + Hybrid Framework
* Maven
  + Introduction & Advantages
  + Configuration
  + Adding dependencies
  + Executing tests
  + Extent Report
* Cucumber
  + Introduction to BDD Approach
  + Advantages
  + Configuration
  + Feature File
  + Keywords in feature file
    - Feature
    - Scenario
    - Given
    - Then
    - When
    - And
    - Background
    - Examples
    - Scenario Outline
  + Step Definition
  + Runner class
  + Tags in Cucumber
  + Hooks in Cucumber
  + Data Driven Testing in Cucumber
* Revision of Java
  + Language Basics
    - Variables
    - Operators
    - Checking conditions
    - Loops (for loop and for each loop)
    - Functions
    - Arrays
  + Class & Object
    - Creating Class
    - Creating object
    - Constructor
    - Static members and functions
  + Inheritance
    - Extends keyword
  + **Interface**
  + Exception Handling
    - **throws** keyword
  + Collections
    - **List**
    - Set
    - Map

**Software Testing**

It is the process of checking Correctness, Completeness, Security & Quality of developed software application.

**Manual Testing Process**

1. Creating Test Scenarios
2. Creating Test Cases
3. Creating Test Data
   1. ECP
   2. BVA
   3. EG
4. Creating RTM

**Operations involved in Testing**

1. Entering data in the text box
2. Clicking on buttons
3. Navigation from one page to another
4. Selecting value from Check box, Radio button, Dropdown list, list box
5. Navigation from tabs
6. Validating the expected result with actual result
7. Generating report
8. Mark the test case Pass or Fail

In manual testing tester uses their **Hand-Eye-Brain** Coordination.

**What is Automation Testing?**

* Perform all above actions to test any application via a machine.
* Machine in this context is Test Automation Tool / Automation Testing Tool
* Every tool is a software
* Every tool understand specific programming language
* You have to give the instructions using any one of the supported programming language.

**Advantages & Need of Automation**

* Repetition
* Saves time
* Faster executing
* Reduces human efforts
* Regression Testing
* Less human errors
* Reusability
* Improve the quality
* 100% Test Coverage
* Compability Testing
* Huge amount of data can be tested

**When to Automate and which tests to be automated?**

* Stable build
* Repetitive task
* Large data
* Compability Testing
* Performance Testing
* Frequent Regression testing
* Security testing

**Types of Tools**

* Unit Testing
  + JUnit
  + NUnit
* GUI Testing / Functional Testing
  + Selenium WebDriver
  + Tosca
  + QTP
  + Appium
  + Playwright
* API Testing
  + Postman
  + REST API

**Process of Automation**

1. Planning
2. Selection of Tool
   1. Type of application to be tested. (AUT / SUT)
   2. Cost of tool
   3. Support availability
   4. Tool availability / market presence
   5. Testers availability
3. Creating Test Script
4. Creating Test Data
5. Execution
6. Report
7. Maintance

**Selenium**

It is suite / bundle of Test Automation Tools to test Web Based Applications (Web Sites)

**Components of Selenium**

1. Selenium IDE – Record and Playback
2. Selenium Grid – Parallel Execution
3. Selenium RC -
4. Selenium WebDriver

**Selenium WebDriver**

* It is an automation testing tool to test web based applications
* It is an API (Application Programming Interface)
* Interface in Java

**Pre-Requisite for Selenium WebDriver**

1. Minimum Windows 10
2. Java (Minimum JDK 11)
3. Editor (IDE)
   1. Eclipse
   2. Idea Itellij
4. At least one Updated Browser
5. Selenium Jar File

**Configuration**

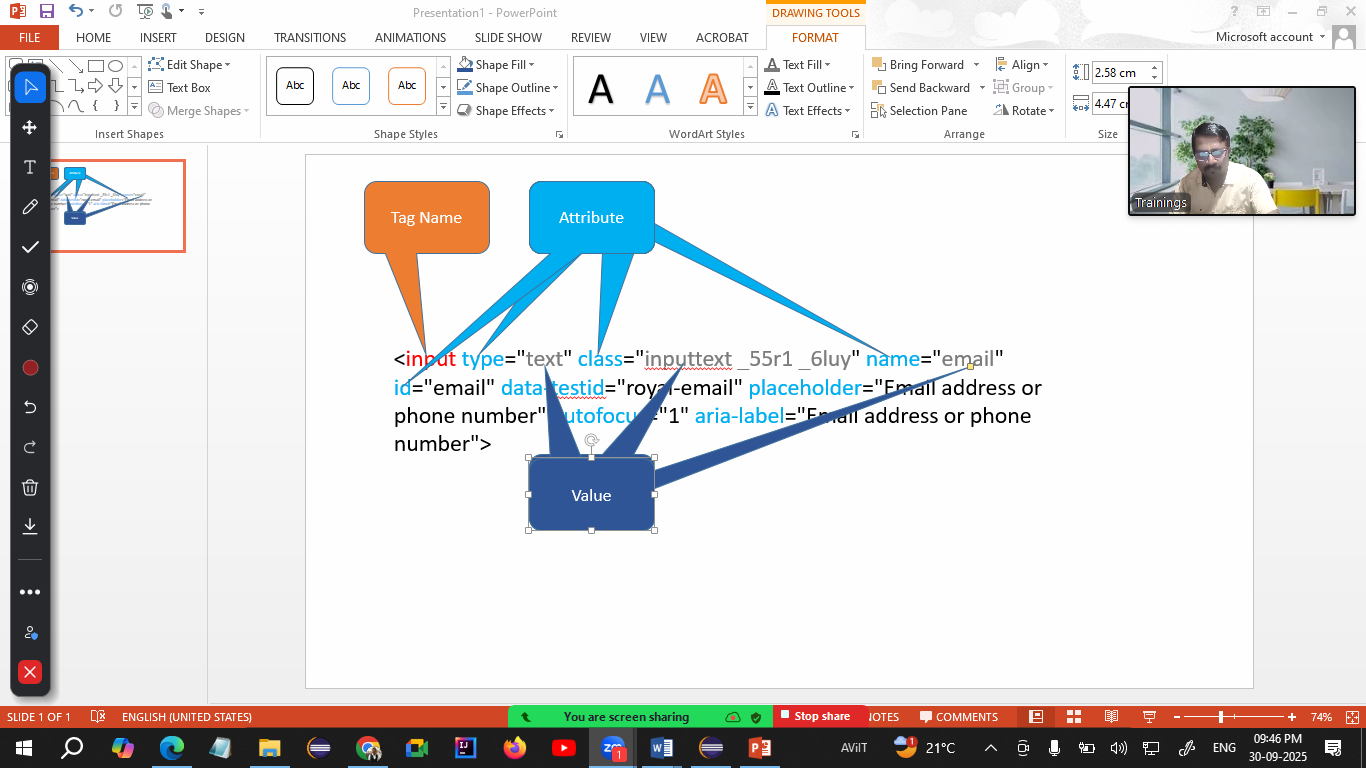
1. Create 2 folders
   1. YourName\_SeleniumDemos (Gaurav\_SeleniumDemos) 🡪 To store all selenium demos
   2. Selenium Jar Files 🡪 To store all the jar files (libraries) required for Selenium
2. Launch <https://www.selenium.dev/>
3. Click on Downloads link
4. To download the Selenium Jar file Click on Latest stable version [4.35.0](https://github.com/SeleniumHQ/selenium/releases/download/selenium-4.35.0/selenium-server-4.35.0.jar)
5. Open your Downloads folder (or where this file is downloaded)
6. Copy this file and paste in 2nd folder (Selenium Jar Files)
7. Open Eclipse
8. Click on Browse button and select the 1st folder which you have created in earlier steps.
9. Click on Launch button (Here we selected the workspace ie. Folder)
10. Click on File 🡪 New 🡪 Java Project
    1. Give the name for project (SeleniumAutomationProject)
    2. Select the Java version (only of it is showing less that 11)
    3. Uncheck **Create module-info.java file** checkbox
    4. Click on Finish
11. Create a package in this project. (com.WebDriverDemos)
12. Create a class in this package.  
    **Selenium Configuration**
13. Right click on Project 🡪 Build Path 🡪 Configure Build Path
14. Click on Libraries tab
15. Click on Classpath
16. Click on Add External JARs…
17. Select the file which you downloaded and stored in 2nd folder
18. Click on Open button
19. Click on Apply and Close button

**WebDriver Methods.**

1. Create object of WebDriver interface 🡪 Launch the blank browser window.
2. get() 🡪 Open the URL.
3. driver.manage().window().maximize() 🡪 Will maximize the browser window.
4. close() 🡪 Will close the **current** browser window.
5. getTitle() 🡪 Will return the title of web page opened by WebDriver object. (String)
6. getCurrentUrl() 🡪 Will return the URL of web page. (String)
7. getPageSource() 🡪 Will return page source (rendered HTML) of page (String)
8. findElement() 🡪 It will find and return single control on the page. **Always locate for the first occurrence.** (WebElement)
9. findElements() 🡪 Will return multiple controls on the page. (List<WebElement>)
10. getWindowHandles 🡪 will return id’s / name of all the browser windows those are opened by WebDriver object. (Set<String>)
11. quit() 🡪 will close all browser windows those are opened by WebDriver object

**Common Exceptions**

1. InvalidArgumentException 🡪 The URL is not in the correct format. URL should be absolute means should start with https / http
2. NoSuchElementException 🡪 Selenium is unable to locate / find the control due to
   1. Your locator value may be wrong.
   2. Your locator value may be dynamic.
   3. Synchronization issue
   4. The control may be in a iframe.
3. InvalidSelectorException 🡪 The value of locator is not in the correct format.
4. SessionTimeoutException 🡪 If the page is not getting loaded within 30 seconds. Because driver.get() is having implicit wait of 30 seconds by default.
5. NoAlertPresentException 🡪 You are trying to handle an alert but there is no alert present on the page.



**Locators**

Locators are the way / method to locate or identify the controls on the page.

1. Name
2. Id
3. ClassName
4. CssSelector
5. LinkText
6. PartialLinkText
7. XPath
8. TagName

**WebElement**

* Every control on the web page is treated as WebElement like Text box, command button, radio button, hyperlink, dropdown list, list box, some text on the page etc.
* WebElement is an interface in Selenium. Which can hold any control on the page.

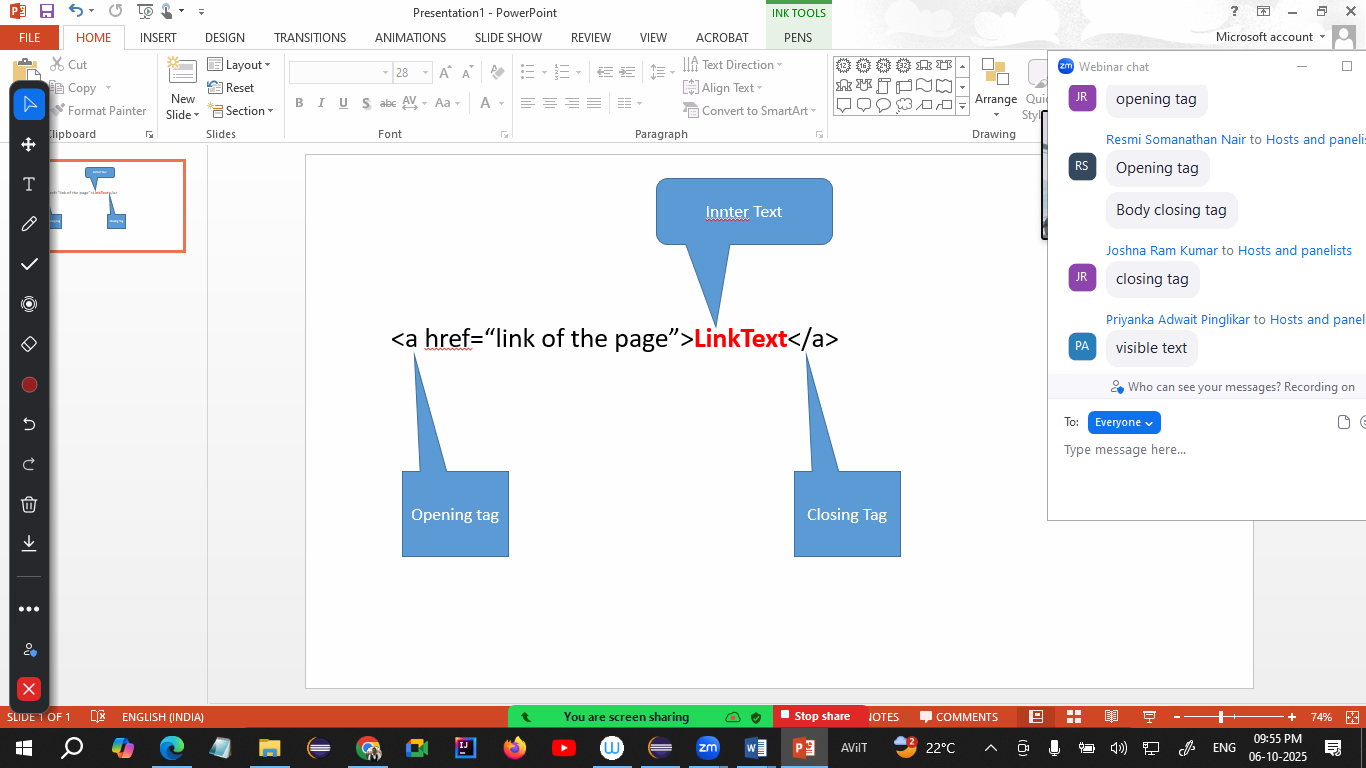
**Methods of WebElement Interface**

1. sendKeys() 🡪 Will enter the text in text box. Will append the text in the text box.
2. click() 🡪 Will click on any control.
3. getText() 🡪 Return the text on the control. (String)
4. isSelected() 🡪 Will check that whether the control (check box / radio button) is selected or not. (boolean)
5. isEnabled() 🡪 Will check that whether the control is enabled or disabled (boolean)
6. isDisplayed() 🡪 Will check that whether the control is visible or not (boolean)
7. getAttribute() 🡪 Will return the value of any attribute of control (String)
8. getLocation() 🡪 Will return the location of the control, x & y co-ordination (Point)
9. getSize() 🡪 Returns the size of WebElement in pixels (Dimention)

**CssSelector**

You can read / find any control on the page using any one of the attribute of that control.

1. Using Single Attribute  
   Syntax:   
   tagName[attribute=”value”]  
   input[placeholder="Email address or phone number"]
2. Using Multiple Attributes  
   Syntax  
   tagName[attribute1=”value”][attribute2=”value”]
3. Using Special Characters
   1. ^ 🡪 Starts With (If starting characters are fixed and ending characters are dynamic)
   2. $ 🡪 Ends With (If ending characters are fixed and starting characters are dynamic)
   3. \* 🡪 Contains text (Checks for the text containing in the attribute value)



**ClassName Locator**

If your control is having multiple classes applied (multiple classes are separated by space) then pick any one of the class name to locate the control.

Pattankodoli Bus stand 🡪 Take a right turn 🡪 Hupare Nagar 🡪 Water Tank 🡪 Lane No 9 🡪 House No 1128 (Ankush Home)

**XPath (XML Path)**

1. Absolute XPath  
   Always starts with html  
   html/body/div/div/div/div/div/div/div[2]/div/div/form/div/div/input
2. Relative XPath  
   Starts with //
   1. Taking reference of parent tag
   2. Taking reference of the exact tag”

**Handling Dropdown List or List Box**

* If any one of the control is having <select> tag, then only Selenium treats such control as drop down list or list box.
* For handling drop down list or list box Selenium has provided special class called as **Select**

Operations

* Show selected country
* Show total no of countries
* Show list of all the countries
* Select Netherland
* Show selected country

Methods of Select Class

1. getFirstSelectedOption() 🡪 Will return the option that is selected from drop down list. (WebElement)
2. getOptions() 🡪 Will return list of all the options / elements from drop down list (List<WebElement>)
3. selectByVisibleText() 🡪
4. selectByContainsVisibleText() 🡪
5. selectByValue() 🡪
6. selectByIndex() 🡪
7. getAllSelectedOptions() 🡪 Will return list of all the selected options / elements form the list box (List<WebElement>)
8. isMultiple() 🡪 Checks that whether the control allows to select multiple options or not (boolean)
9. deselectByIndex()
10. deselectByVisibleText()
11. deselectByContainsVisibleText()
12. deselectByValue()
13. deselectAll()

**Synchronization (Waits in Selenium)**

It is the process of adjusting speed of tool with speed of application.

1. Thread.Sleep() 🡪 Will pause the execution of script for specified milliseconds.
   1. Takes mandatory delay
   2. It is applicable to single statement only
2. ImplicitWait
   1. It doesn’t take mandatory delay
   2. It is applicable throughout the script
3. ExplicitWait
   1. It is applicable to single statement only
   2. It doesn’t take mandatory delay
   3. You can handle some conditions like element to be clickable, alert to be present, element is visible etc.
4. FluentWait
   1. It is applicable to single statement only
   2. It doesn’t take mandatory delay
   3. You can handle some conditions like element to be clickable, alert to be present, element is visible etc.
   4. It can handle exception as well.
   5. Parameters for method  
      w - withTimeout  
      i - ignoring  
      p - pollingEvery  
      u - until
5. PageLoadTimeout 🡪 you can add more time to load the page to avoid SessionTimeoutException.

**Handling Table**

1. Display all headers
2. Display total no of rows
3. Display any row randomly

**JavascriptExecutor Interface**

* Used to Scroll the page horizontally, vertically or both
* Used to click on the control which is hidden by some another control.

**Handling Alert**

Alert is something

* That is not able to inspect
* Not having any (x) close button
* If Alert is present you cannot perform any action on the page.
* Actions can be performed only after handling this alert

For handling such alerts, Selenium has given a special interface called as **Alert Interface.**

**Methods of Alert interface**

1. driver.switchTo().alert() 🡪 Will switch to the alert
2. getText() 🡪 will return text on the contrl. (String)
3. accept() 🡪 will click on Ok button
4. dismiss() 🡪 will click on Cancel button
5. sendKeys() 🡪 Will enter the text on Prompt Box

**Handling File Upload**

If the control is having **type=”file”** attribute then simply call **.sendKeys()** method to upload the file.

Pass the path of the file to be uploads

**Mouse Actions**

1. Hover
2. Click (Left)
3. Drag and Drop
4. Double Click
5. Right Click

**Actions** class is used to perform all above actions.

Methods

1. moveToElement() 🡪 Hover the mouse on the control
2. perform() 🡪 Will perform the said action on the control
3. contextClick() 🡪 will do Right Click on the control
4. doubleClick() 🡪 Will click Double Click on the control

TestNG (Test Next Generation)

It is testing framework, which makes your automation testing easy.

**Framework:** Is the set of rules, guidelines to test your application in easy way.

**Features of TestNG**

* Allows to create multiple tests in one class
* Allows to set priority to the tests
* Uses annotations
  + @Test
  + @BeforeTest
  + @AfterTest
  + @BeforeMethod
  + @AfterMethod
  + @BeforeClass
  + @AfterClass
* Generates a Report
  + Normal report
  + HTML Report
* Data Driven Testing using @DataProvider
* Allows to execute / skip single / multiple tests
* Allows Parallel Execution
* Allows to execute multiple classes, package
* Parameterization
* Allows to implement variety kind of frameworks
  + Modular Framework
  + Keyword Driven Framework
  + Data Driven Framework
  + Page Object Model
  + Hybrid Framework

**Annotations**

1. @Test 🡪 This method is treated as a test case
2. @BeforeTest 🡪 This is the method that get executed **only once before executing 1st test case**
3. @AfterTest 🡪 This is the method that get executed **only once after executing last test case.**
4. @BeforeMethod 🡪 This is the method that get executed **before every test case**
5. @AfterMethod 🡪 This is the method that get executed **after every test case**
6. @DataProvider 🡪 This is the method that sends the data to @Test method, and @Test method will get executed for multiple times

**Points to be noted about Configuration methods**

1. They need not to be in a pair
2. They can appear anywhere in the code / sequence does not matter.

BeforeTest

BeforeMethod

Test 1

AfterMethod

BeforeMethod

Test 2

AfterMethod

BeforeMethod

Test 3

AfterMethod

BeforeMethod

Test 4

AfterMethod

AfterTest

**Data Driven Testing**

Executing single test for multiple times with multiple data

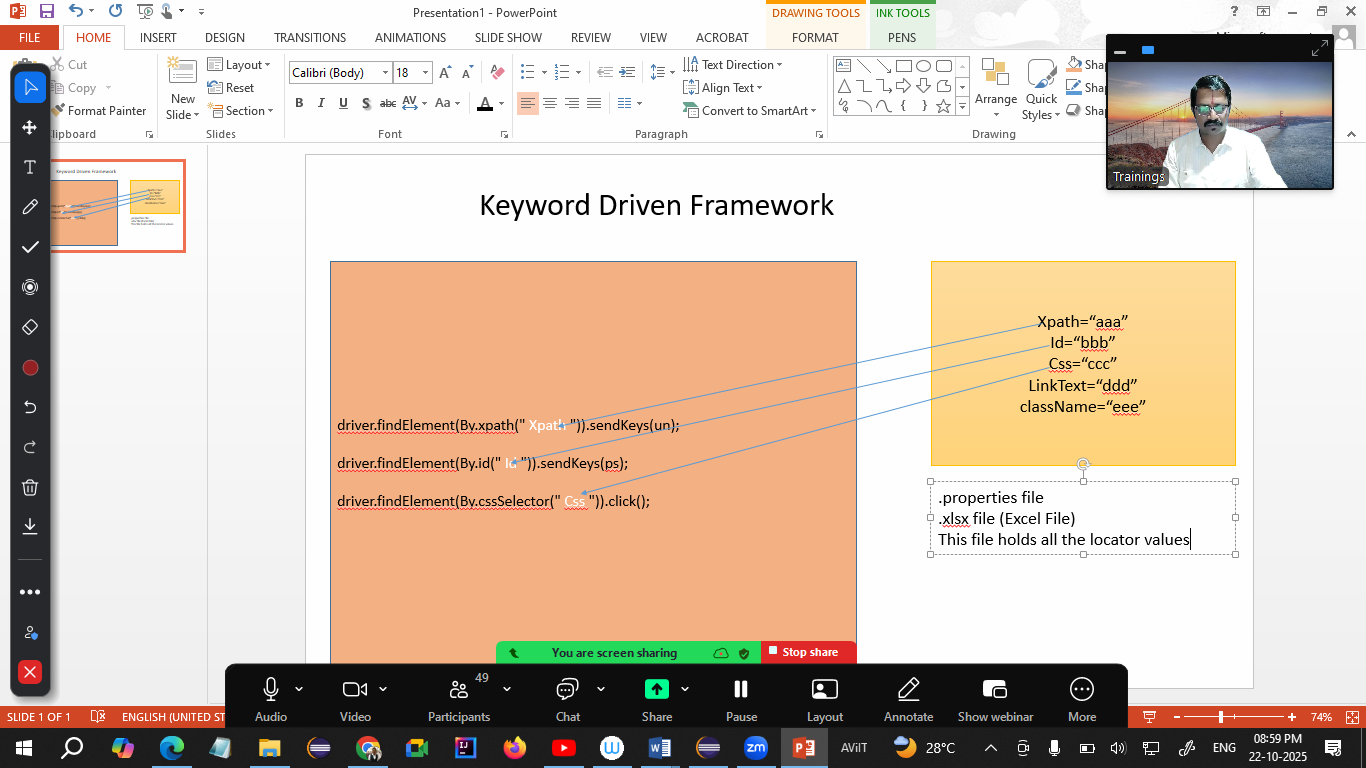
**Assertion in TestNG**

Assertion is the way of marking any test case as pass or fail. This is the process of validation.

**Execution Via XML File**

* TestNG allows you to execute / skip single test / multiple tests
* This is possible via .xml file
* **You can execute only TestNG classes via .xml file**
* While creating .xml file please note that
  + All the tags are pre-defined
  + Don’t alter (change the sequence of) tags

**Keyword Driven Framework**



**Data Driven Framework**

Reading data from Excel file

Writing data to Excel

Download [poi-bin-5.2.3-20220909.zip](https://archive.apache.org/dist/poi/release/bin/poi-bin-5.2.3-20220909.zip)

Copy and paste in the 2nd folder (Selenium Jar File)

Unzip this file

You can see total 21 files in this folder.

You have to add all these 21 files in your project

1. Right click on Project 🡪 Build Path 🡪 Configure Build Path
2. Click on Libraries tab
3. Click on Classpath
4. Click on Add External JARs…
5. Select all 21 files one by one folder which you extracted
6. Click on Open button
7. Click on Apply and Close button

**Maven**

* It is build management tool
* Product of Apache group
* Developer and tester both can use Maven
* Only single project can be used by both developer and tester
* Use dependency for configuration instead of adding the jar file.
* Easy configuration via pom.xml file (Project Object Model)

**Creating Maven Project**

1. File 🡪 New 🡪 Maven Project
2. Select the first checkbox (Create a simple project skip archetype selection)
3. Click on Next
4. Given the group id and artifact id (Both should be same)
5. Click on finish

**BDD – Behavior Driven Development**

* BA, Developer & QA team will gather the requirements.
* These requirements will be shared with QA team
* QA Team will create **Feature File (1st component of BDD)**
* Feature file contains the test scenarios
* This feature file will be shared with
  + Developer
  + Manual Tester
  + Automation Tester
* Now automation tester will execute this feature file and they will get **Step Definition / Glue Code (2nd Component of BDD)**
* Automation tester will write the script in this glue code
* **Runner Class (3rd Component of BDD)** is required for execution of the script

**BDD can be implemented using a tool called as Cucumber**

**Creating Cucumber Project**

1. File 🡪 New 🡪 Maven Project
2. Select the 3rd checkbox (Add project to Working Set)
3. Click on Next
4. Add **io.cucumber** in filter box
5. Click on the archetype you are getting
6. Click on next
7. Give Artifact Id & Group ID
8. Click on Finish
9. Open pom.xml file
10. Select the code from <properties> till </dependencies> and delete this code
11. Open <https://github.com/cucumber/cucumber-java-skeleton/commit/d7249b50c570816eba27ce94557e1de7e9b0f97>
12. Select the code from <properties> till </dependencies> (line no 11 to 41)
13. And paste in the pom.xml file where you have deleted the code in earlier step.
14. Save this file
15. Change the Java version from the pom.xml file  
    <java.version>1.8</java.version>

Delete following lines  
<cucumber.version>6.8.2</cucumber.version>  
<maven.compiler.version>3.8.1</maven.compiler.version>

<maven.surefire.version>2.22.2</maven.surefire.version>

**Feature File**

* Collection of test scenarios
* Having extension .feature
* Created using Gherkin language / Gherkin syntax
* Contains keywords
  + **Feature**: Represents the requirement
  + **Scenario**: Represents the test condition / test scenario to be tested
  + **Given** Represents pre-requisite
  + **When** Represents the steps to be performed (For every step you can have separate When statement)
  + **And** Used to combine multiple When statements
  + **Then** Represents Expected Result
  + **Background**: Can be used when you are having multiple scenarios and common Given statement
  + **Examples**: Represents the data to be passed for data driven testing (Like @DataProvider in TestNG)
  + **Scenario Outline**: Is used while Data Driven Testing

Feature File

Requirement: Validate title of Google

1. Open Google
2. Read the title
3. Validate that title should be Google

Feature File

Feature: Google Title

Scenario: To validate title of Google

Given Open Google

When Read the title of page

Then Title should be Google

Requirement: Google Search Functionality

1. Open Google
2. Type the keyword to be searched
3. Hit Enter
4. A valid search should display

Feature File

Feature: Search Functionality on Google

Scenario: To validate search functionality on Google

Given Open Google

When Enter keyword for searching

And Hit Enter

Then A valid search result should display

**Tags in Cucumber**

* Used to execute or skip single / multiple scenarios
* Tags needs to be specified in feature file
* Tags starts with @
* Execution criteria needs to be specified in Runner class

**Hooks in Cucumber**

* These are special methods those get executed before and after every scenario (Like @BeforeMethod & @AfterMethod in TestNG)
* There are two hooks in cucumber
  + @Before – Executed before every scenario
  + @After – Executed after every scenario