Trainer: Ankush Vankore

Selenium WebDriver (3rd Module)

Contents

* Concept of Automation
* Components of Selenium
* Configuration of Selenium WebDriver
* Methods of WebDriver
  + get()
  + getCurrentUrl()
  + getPageSource()
* Locators in WebDriver
  + Name
  + ID
  + ClassName
  + CssSelector
    - Single Attribute
    - Multiple Attributes
    - Special Characters
  + Xpath
    - Absolute Xpath
    - Relative XPath
  + LinkText
  + PartialLinkText
  + TagName
  + Relative Locator
* Handling the Controls
  + Button
  + Textbox / Text Area
  + Hyperlink
  + Radio button
  + Checkbox
  + Dropdown list
  + File Upload
  + Tables
* Synchronization (Waits in Selenium)
  + Thread.Sleep()
  + Implicit Wait
  + Explicit Wait
  + FluentWait
  + PageLoadTimeout
* Handling Alerts
* Handling Multiple Windows
* Handling Popups
* Mouse Actions
  + Click
  + Right Click
  + Drag and Drop
  + Mouse Hover
* JavascriptExecutor
  + Click
  + Scroll
* Screenshot
* TestNG (Framework)
  + Configuration
  + Executing Single test case
  + Executing Multiple test cases
  + Setting priority
  + Getting the Reports
  + BeforeTest, AfterTest, BeforeMethod, AfterMethod etc
  + DataProvider
  + Execution via XML file (Skipping or executing tcs) / Modular Framework
  + Keyword Driven Framework
  + Page Object Model
  + Data Driven Framework / Testing
* Introduction to Maven
  + How to create a project
  + Adding dependencies
  + Execution
  + Extent Report
* BDD Framework (Cucumber)
  + Configuration
  + Components of Cucumber
    - Feature File
    - Step Definition
    - Runner Class
  + Keywords
    - Feature
    - Scenario
    - Given
    - When
    - Then
    - And
    - Background
    - Examples
    - Scenario Outline
  + Tags in Cucumber
  + Reports in Cucumber
  + Hooks in Cucumber
  + Data driven testing in Cucumber

Java Revision Points

* Java Basics
  + Variables
  + if else/
  + loops
    - for loop
    - enhanced for loop (for each loop)
    - nested for loop
  + Arrays
  + Methods
  + Type casting
* Creating class and objects
  + Constructor
  + Static variables and methods
  + Calling methods with objects
* Exception Handling
  + throws keyword
* Collections
  + List (VVVVVVVVVVIMP)
  + Set
  + Map

What is Software Testing?

Software Testing is a process of identifying Correctness, Completeness, Security and Quality of developed software application. (CCSQ)

Scenario Creation

Test Case Creation

Test Data Creation

RTM Creation

Execution

Manual Testing Actions

* Entering the data
* Selecting the data from drop down list
* Clicking on buttons / links
* Selecting radio button / check box
* Observe the actual result
* Comparing Actual result v/s expected result
* Marking test case as pass or fail

Automation testing is the process via which we can perform all above actions via a machine.

**The machine in this context is Automation Testing Tool / Test Tool / Testing Tool / Test Automation Tool.**

Every automation tool is a software.

Instructions needs to be provided in specific programming language (Java, Python, Javascript, C# .NET)

**Advantages of Automation Testing**

* Faster
* Time saving
* Cost
* Less human efforts
* Accuracy
* Reusability
* Automated Report
* More Efficiency
* Improve the Quality
* More Test coverage
* Early bug finding
* Testing on multiple platform
* Boost the productivity

**Need**

* Speed
* 100% Test Coverage
* CICD
* Automated Reports

**When to Automate**

* Stable Requirement
* Regression Testing
* Repeated Scenarios
* Large amount of Data
* Performance Testing
* Security Testing
* Compability Testing

**Which test cases can be automated?**

* Regression
* Repetitive
* Performance

**Types of Automation Testing Tools**

* Unit Testing
  + JUnit, NUnit
* API Testing
  + Postman, REST API
* Performance Testing
  + JMeeter, Load Runner
* Functional Testing
  + Selenium, Tosca, QTP

**Process of Automation**

* Planning
* Tool Selection
  + Type of application
  + Cost of tool
  + Easiness of tool
  + Testers Availability
  + Automated Report
  + Community support
  + Market presence
* Test script creation (Code for automation)
* Test Data Creation
* Execution
* Report Creation
* Maintance

Selenium

Set or Bundle of Test Automation tools

Components of Selenium

1. Selenium IDE (Record and Playback mechanism)
2. Selenium Grid (Parallel Execution)
3. Selenium RC (Deprecated)
4. Selenium WebDriver

Selenium WebDriver

1. Automation Testing Tool for testing web based (browser based / Web Site) application.
2. WebDriver is an interface in Java.
3. An API.

Pre-Requisite for Selenium WebDriver

1. Minimum Windows 10
2. Any one UPDATED browser
3. Minimum Java 11 (For checking fire **java –version** command on command prompt)
4. Any Java Editor
   1. Eclipse
   2. Intellij
5. A Selenium Jar file

Configuration of Selenium WebDriver

1. Create two folders
   1. YourName\_SeleniumDemos
   2. Selenium Jar Files
2. Open selenium.dev website
3. Click on Downloads tab (Link)
4. Click on Latest stable version X.XX.XX (Link)
5. Cut paste the downloaded file in the 2nd folder which we have created in 1st step.
6. Launch Eclipse
7. Select the 1st folder (Which we have created in 1st step) as your workspace.
8. Click on Launch
9. Now create a new Project
   1. File 🡪 New Java Project
   2. Give a name to project
   3. Uncheck **Create module-info.java file** checkbox
   4. Click on Finish (This will create your project)
   5. Create a package and class in the package.
10. Now configure Selenium WebDriver to your project
    1. Right click on your project
    2. Go to Build Path
    3. Click on Configure Build Path
    4. Click on Libraries Tab
    5. Click on Classpath
    6. Click on Add External JARs… button
    7. Open 2nd Folder and select the file which you have downloaded in earlier steps.
    8. Click on Apply and Close

Methods of WebDriver Interface

1. Launching the browser: Create object of WebDriver interface.
2. get(): Used to open any web site.
3. close(): Used to close the browser window which is opened by WebDriver object.
4. driver.manage().window().maximize(): Will maximize the browser window.
5. getTitle(): Which will return title of the webpage (String)
6. getCurrentUrl(): Returns the URL of current page. (String)
7. getPageSource():
8. findElement(): Reads the control on the web page. Locates for the first occurrence on the web page. (WebElement)
9. findElements(): Reads multiple controls on the web page. (List<WebElement>)

**Possible Exceptions**

1. InvalidArgumentException: Your URL is in wrong syntax. (The URL should be absolute means it should start with http)
2. NoSuchElementException: Selenium is unable to locate that control. Possible reasons area
   1. The value of locator is wrong.
   2. The value of locator may be dynamic.
   3. Synchronization issue
3. InvalidSelectorException: The value of locator is in wrong format.
4. TimeoutException : The specified page is not getting loaded within 30 seconds.
5. ElementClickInterceptedException : Selenium is not able to click on the control my possible the control is hidden by some another control.
6. NoAlertPresentException : There is no alert displayed on the page and you are trying to handle alert.

**WebElement**

* Every control on the web page is treated as WebElement in Selenium.
* WebElement is an interface.
* WebElement can hold any control on the web page.

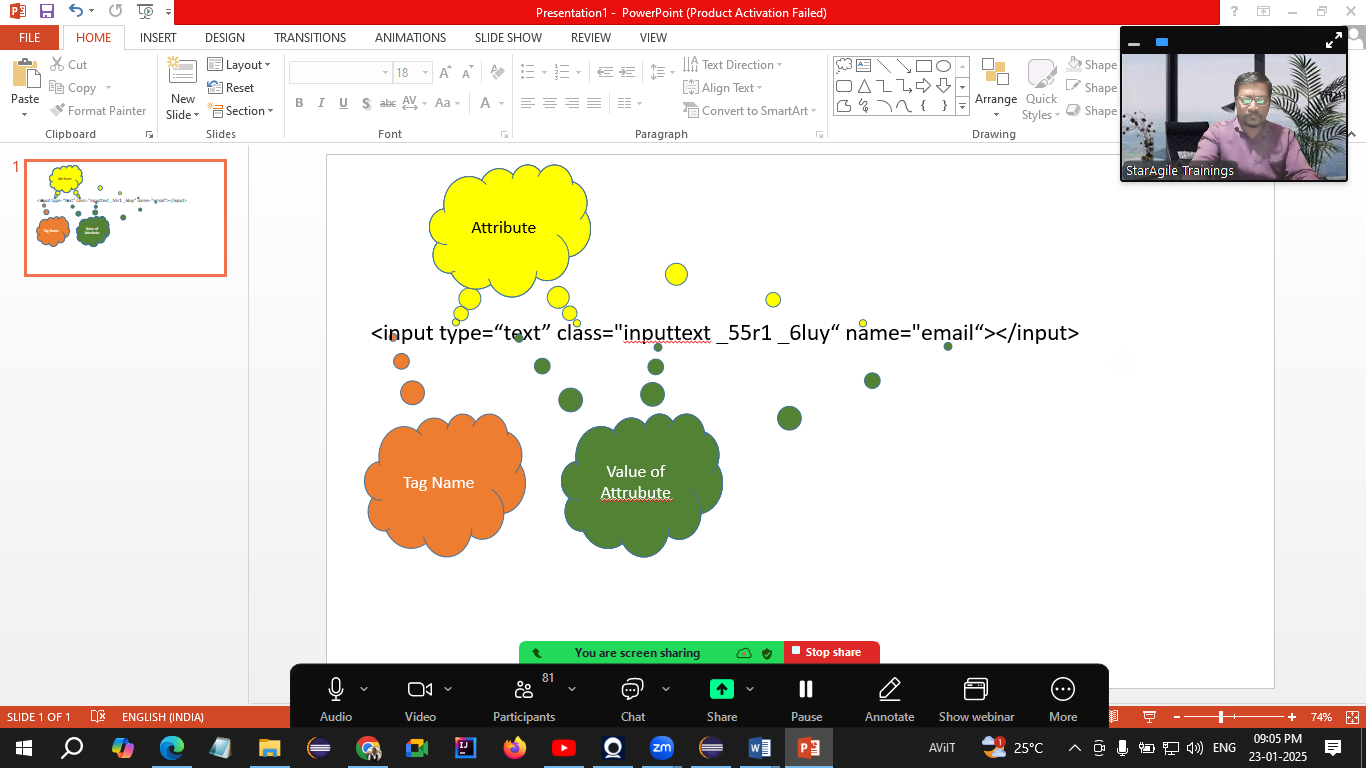
**Methods of WebElement interface**

1. sendKeys(): Used to enter some text in the text box.
2. click(): Used to click on the control.
3. getText(): Returns the text on the control. (String)
4. isSelected() : Used to check whether the checkbox or radio button is selected or not. (boolean)
5. isEnabled() : Used to check whether the control is enabled or disabled. (boolean).
6. isDisplayed() : Used to check whether the control is visible or not. (boolean)
7. getAttribute() : Returns value of any attribute of the control. (String)

**Locators**

Locators are way to find out or read any control (WebElement) on the web page.

1. Name
2. Id
3. className
4. CssSelector
5. Xpath
6. LinkText
7. PartialLinkText
8. TagName
9. RelativeLocator



CssSelector

This is the locator used to read the control using any one or multiple attributes of the control.

Types

* Single Attribute  
  Syntax  
  tagName[attribute=”value”]  
  eg. Input[type="text"]
* Multiple Attributes  
  Syntax  
  tagName[attribute1=”value”][attribute2=”value”]  
  eg. Input[type="text"]
* Special Characters
  + ^ 🡪 Start With
  + $ 🡪 Ends With
  + \* 🡪 Contains
  + .
  + #

**Xpath in Selenium**

XML Path

Pattankodoli (Kolhapur Dist) Bus Stand 🡪 Take right turn 🡪 Hupare Nagar 🡪 Near Water tank 🡪 Lane no 9 🡪 HNo 1128

Types of Xpath

1. Absolute Xpath

Starts with html

1. Relative Xpath

Starts with //

* 1. Using parent tag  
     //parentTag[@attribute=”value”]/tag/tag
  2. Using the exact tag  
     //tagName[@attribute=”value”]

**Handling Dropdown List**

If the control is having <select> tag then only Selenium treats it as a dropdown list.

For handling such a drop down lists, Selenium has given a special class called as Select class.

Select Class 🡪 For handling dropdown list or list box

Methods

1. getFirstSelectedOption() : Returns the selected option from the drop down list. (WebElement)
2. getOptions() : Returns list of all the options from dropdown list. (List<WebElement>)
3. selectByVisibleText() : Selects the option by its inner text / text on the option.
4. selectByValue() : Selects the option by using it value attribute.
5. selectByIndex() : Selects the option by using its 0 based index.
6. getAllSelectedOptions() : Returns list of all the selected items from list box. (List<WebElement>)
7. isMultiple() : Checks whether the control is list box or dropdown list. Returns true if it is List Box (boolean)
8. deselectByIndex() :
9. deselectByValue() :
10. deselectByVisibleText() :

Operations

* Display name of selected country.
* Display total no of countries
* Display list of all the countries
* Select Saudi Arabia
* Display name of selected country.

**Synchronization (Waits in Selenium):**

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

1. Thread.Sleep() : Pauses the execution of the script for specified number of milliseconds
   1. It takes mandatory delay.
   2. It is applicable to single statement only.
2. ImplicitWait
   1. Doesn’t take mandatory delay.
   2. It is applicable throughout the script.
3. ExplicitWait (WebDriverWait)
   1. Doesn’t take mandatory delay.
   2. It is applicable to single statement only.
   3. You can put the condition for wait.
4. FluentWait
   1. Doesn’t take mandatory delay.
   2. It is applicable to single statement only.
   3. You can put the condition for wait.
   4. We are able to handle any specific Exception.

Parameters for FluentWait  
w - withTimeout  
i – ignoring (Exception)

p - pollingEvery

u - until

1. PageLoadTimeout
   1. Used to avoid TimeoutException which is caused due to page is not getting loaded within 30 second.
   2. You can add some more seconds for loading the page.

**Handling Table**

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

**Handling JavaScript Alert**

Alert is an interface which is used to handle any alert.

Methods

1. driver.switchTo().Alert() : Will switch to the alert.
2. getText() : Will return the text on alert. (String)
3. accept() : Will click on Ok button.
4. dismiss() : Will click on Cancel button.
5. sendKeys() : Will enter the text on alert (Prompt box / input box)

JavascriptExecutor : Is an interface via which we can perform actions via JavaScript.

**Mouse Actions**

1. Mouse Hover
2. Right Click
3. Left Click
4. Drag and drop
5. Double click

**Actions** class in Selenium helps you to perform all above mouse actions.

**TestNG** (Next Generation)

One of the popular testing framework

**Framework:** Set of **rules**, classes, interfaces, APIs for execution of your automation script. Which will make your automation testing more easier.

Advantages

1. Makes automation testing easier
2. You can make collection of multiple test cases
3. You can set priorities to test cases
4. Uses annotations
   1. @Test
   2. @BeforeTest
   3. @AfterTest
   4. @BeforeMethod
   5. @AfterMethod
   6. @DataProvider
   7. @Parameters
   8. @BeforeClass
   9. @AfterClass
5. Allows to generate automatic report
   1. Normal report
   2. HTML report
6. Allows to implement different frameworks
   1. Linear Framework
   2. Modular Framework
   3. Data Driven Framework
   4. Page Object Model
   5. Keyword Driven Framework
   6. Hybrid Framework
7. Allows to execute / skip single / multiple test cases
8. Allows to create and execute tests cases via groups
9. Reusability of script

**Annotations**

1. @Test: This method is treated as Test Case
2. @BeforeTest: This method get executed **only once before executing 1st test case**
3. @AfterTest: This method get executed **only once after executing last test case**
4. @BeforeMethod: This method get executed **before every test case.**
5. @AfterMethod: This method get executed **after every test case**.
6. @DataProvider: This is the method that passes the data to test case.

**Points to be noted about configuration methods**

1. They are getting executed irrespective of its sequence.
2. They need not to be in pair

BeforeSuite

BeforeTest

BeforeClass

BeforeMethod

testHM

AfterMethod

BeforeMethod

testMyntra

AfterMethod

BeforeMethod

testUrbanic

AfterMethod

BeforeMethod

testAmazon

AfterMethod

AfterTest

**Please note:**

* The test cases will execute in it alphabetic order by default.
* If you want to set some specific order for test cases then you can set like @Test (priority = <int no>)
* If two or more test cases are having same priority, then those test case will get executed in its alphabetical order.

Launch the browser (Only once before 1st test case)

Test 1 – Login

Test 2 – Logout

Close the browser (Only once after last test case)

**Data Driven Testing**

Executing single test case with multiple data set.

1. Via Array (2D Array) @DataProvider
2. Via XML file
3. Via Excel file

**Assertion: This is the process via which you can mark test case as pass or fail**

Types of Assertion

1. Hard Assertion: Will stop the execution
2. Soft Assertion: Will throw the exception and continue with execution.

**Modular Framework**

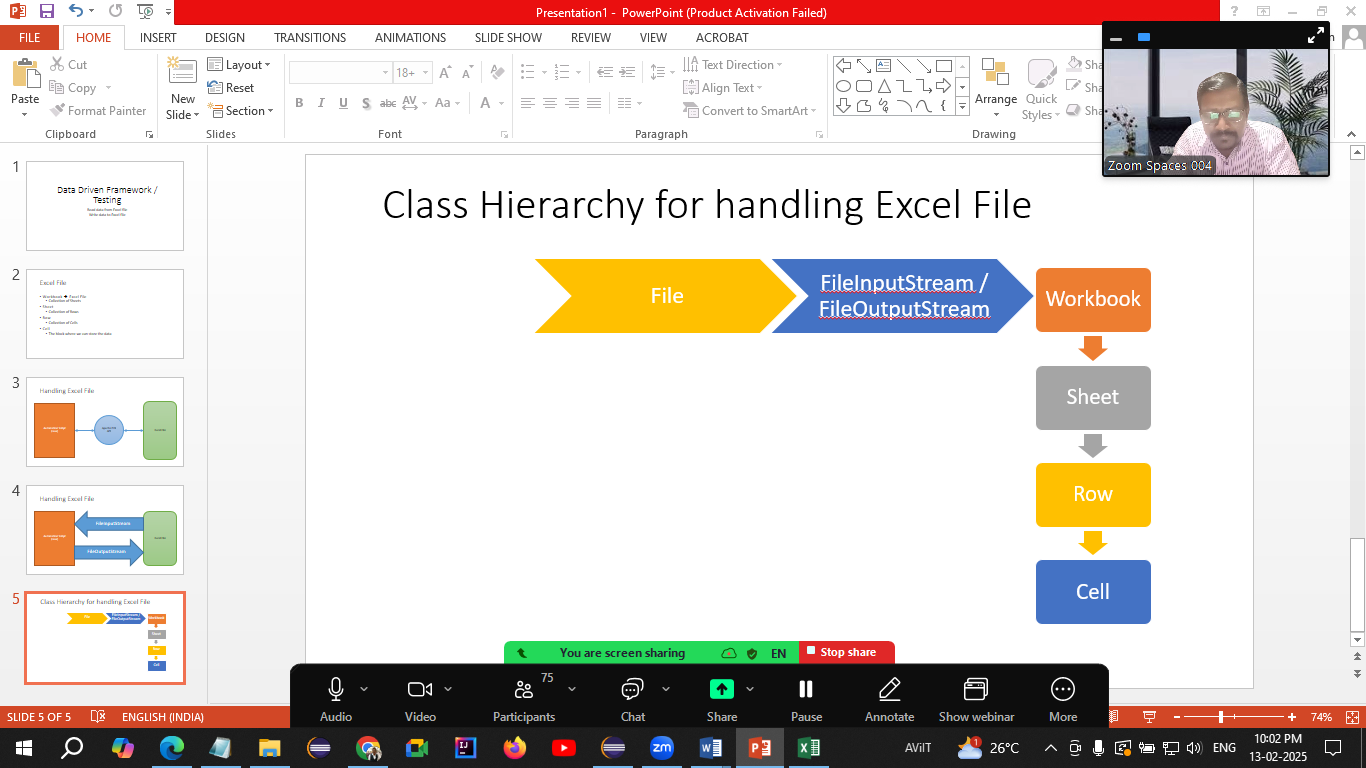
* You can execute or skip single / multiple test cases.
* Execution is done via XML file

**Before creating XML please note that**

1. All the tags are pre-defined
2. You cannot change the sequence of any tag
3. All the tags are case sensitive.

Page Factory:

* It is subset of Page Object Model.
* This is



Link for Apache POI

<https://archive.apache.org/dist/poi/release/bin/>

download last 4th file with name

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

Steps for configuration of Apache POI to Eclipse

* Download this file
* Copy this file to the 2nd folder which we created in 1st lecture (Selenium Jar Files)
* Extract this .zip file
  + observe the folder structure
  + 8 jar files in root folder
  + 6 jar files in lib folder
  + 7 jar files in ooxml-lib folder
* Open Eclipse
* Right click on project 🡪 Build Path 🡪 Configure Build Path 🡪 Libraries 🡪 Class Path 🡪 Add External Jars…
* Add all above 21 files from one by one folder
* Click on Apply and close

Maven:

* Build management tool.
* Single project can be used by both developer and tester
* Developers can use for development purpose
* Testers can use for automation purpose
* Apache product

Advantages

* Version controlling makes simple
* Configuration part is very easy
* Uses dependencies
* Configuration is made via pom.xml file

Creating Maven Project (1st Way)

* File 🡪 New 🡪 Maven Project
* Select 3rd check box (Add Project(s) to working set)
* Next
* Add filter as maven-archetype-quickstart
* Select the latest version
* Next
* Give GroupId and ArtifactId (Both should be same)
  + GroupId is company name
  + ArtifactId is project name
* Finish

Creating Maven Project (2nd Way – Without archetype)

* File 🡪 New 🡪 Maven Project
* Select 1st check box ()
* Next
* Give GroupId and ArtifactId (Both should be same)
  + GroupId is company name
  + ArtifactId is project name
* Finish