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. InvalidSelectorException: The value of locator is in wrong format.

**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.

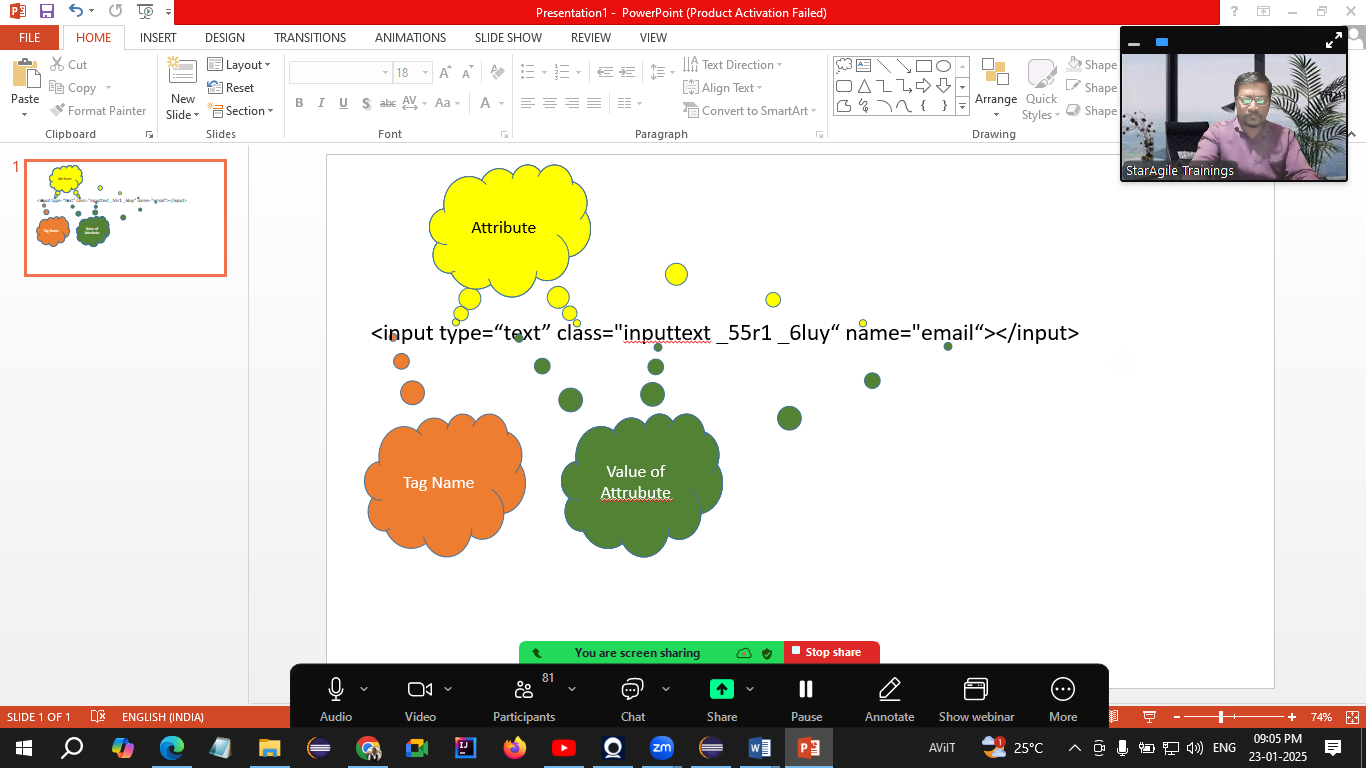
**Methods of WebElement interface**

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)

**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”]