Ankush Vankore

Selenium WebDriver

* Intro
* Configuration
* Locators
* Handling multiple controls
  + Textbox
  + Button
  + Radio button
  + Check box
  + Dropdown list
  + Synchronization / Waits in selenium
  + List box
  + Tables
  + Alerts
  + Mouse Actions
  + Multiple window handling
  + Screenshot
  + Robot class

TestNG

* Introduction
* Configuration
* Executing single test case
* Executing multiple tests
* Assertion
* Annotations
* Frameworks
  + Linear
  + Modular
  + Data driven
  + Keyword driven
  + Page Object model
  + Hybrid

Introduction to Maven

* Intro
* Configuration
* Adding dependencies
* Executing scripts
* Generating reports

BDD Framework using Cucumber

* Intro
* Configuration
* Components
  + Feature file
  + Step Def
  + Runner class
* Keywords in Cucumber
* Tags
* Data driven testing
* Hooks

Java

1. Language Basics
   1. Variables
   2. Data types
   3. Displaying the data
   4. Conditions
   5. Loops (for loop , enhanced for loop)
   6. Arrays
   7. Functions / Methods
2. Class and object
   1. Create class and object
   2. Constructor
   3. Static methods and static members
3. Inheritance
   1. Interface
4. Collections
   1. List
   2. Set
   3. Map
5. Exception Handling
   1. throws keyword

Software Testing:

Process of verifying the **C**orrectness, **C**ompleteness, **S**ecurity and **Q**uality of developed software application.

In manual testing, tester is using hand-eye-brain co-ordanitation.

* Entering some data in text box
* Selecting the data from check box, dropdown list, radio button
* Taking actions, clicking the button
* Navigation from one page to another
* Checking the expected result v/s actual result

Automation means performing all above actions using some automation testing tool. (Testing Tool, Automation Tool)

Every tool is a software.

Instructions needs to be provided using any one of the supporting programming language.

**Advantages of Automation**

* Saves the time
* Reduces the resources
* Reduces the man power
* Reduces human errors
* Useful in regression testing
* Efficiency
* Less manual efforts
* Increase the productivity
* Avoids the repetition
* Cost saving
* Data security
* 100% test coverage
* Increase the quality
* Can be executed from any place
* Consistency
* Accuracy
* Scalability
* Reusability

**Need of automation**

* Quick testing
* Speed
* Avoid repetition
* Cost Saving
* Saves the time
* Maintain Quality
* Reduces human errors
* Testing on different Environment

**When to do Automation?**

* Stable requirements
* Stable build
* Repeated test scenarios
* Regression testing
* Different Environment

**Which TCs should automate?**

* Regression
* Performance
* Cross browser
* API Testing
* Data Driven Testing

**Types of Automation Testing**

* Unit Testing (JUnit / NUnit)
* API Testing (Postman / RestFULL API)
* GUI Testing / Functional Testing (Selenium WebDriver, TOSCA, QTP, Appium, Winium)
* Performance Testing (JMeeter, LoadRunner)

**Tool selection**

* Technology supported
* Budget
* Reports
* Tool availability (Market)

**Process of Automation**

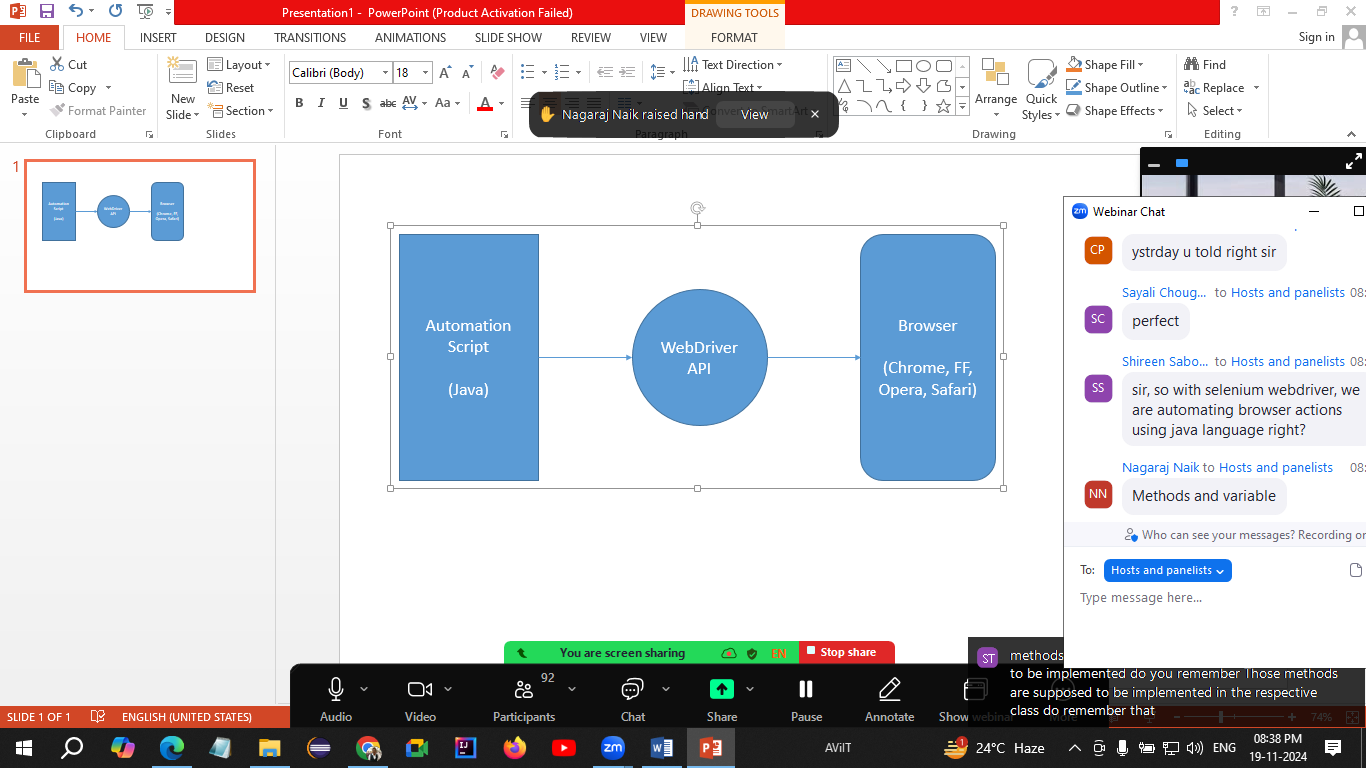
* Planning
* Tool selection
* Creation of the Test Script (Code for automation)
* Test Data creation
* Execution
* Report
* Maintance

**Selenium**

Is the bundle / suite of test automation tools to test WEB BASED APPLICATIONS. (Web Sites)

Components of Selenium

1. Selenium IDE
   1. Record and playback
   2. Generates the script automatically
   3. Very basic level tool.
   4. Doesn’t support for any framework
   5. Reusability is very less
2. Selenium Grid
   1. Parallel testing on multiple browsers, multiple OS, multiple machines
3. Selenium RC (Deprecated)
4. Selenium WebDriver



Selenium WebDriver

* Is an automation tool for testing web based applications.
* An interface in Java.
* An API

Pre-Requisite

* Minimum Java 11 (JDK 11)
* Minimum Windows 10
* Any one latest (Updated) browser
* Any one Editor
  + Eclipse
  + Idea IntelliJ
* Selenium Jar file (Latest)

Configuration of Selenium WebDriver

1. Create 2 folders on D drive
   1. YourName\_SeleniumWebDriverDemos
   2. WebDriver Jar Files
2. Open selenium.dev 🡪 Click on Downloads 🡪 Download Latest Stable Version
3. Open Downloads folder and copy this file to the 2nd Folder which you created in step 1
4. Open Eclipse
5. Select the first folder (YourName\_SeleniumWebDriverDemos) which you have created in step 1 as a workspace
6. Click on Select Folder
7. Click on Launch
8. Create a project
   1. File 🡪 New 🡪 Java Project
   2. Give any meaningful name
   3. Select Environment variable as JavaSE-11 (Minimum)
   4. **Uncheck Create module-info.java file checkbox.**
   5. Click on Finish
9. Create a package inside this project
10. Create a class inside this package
11. Configure Selenium Jar files to the 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…
       1. Open the 2nd folder which you have created in 1st step
       2. Select the Selenium-server file
       3. Click on Open button
       4. Click on Apply and Close button

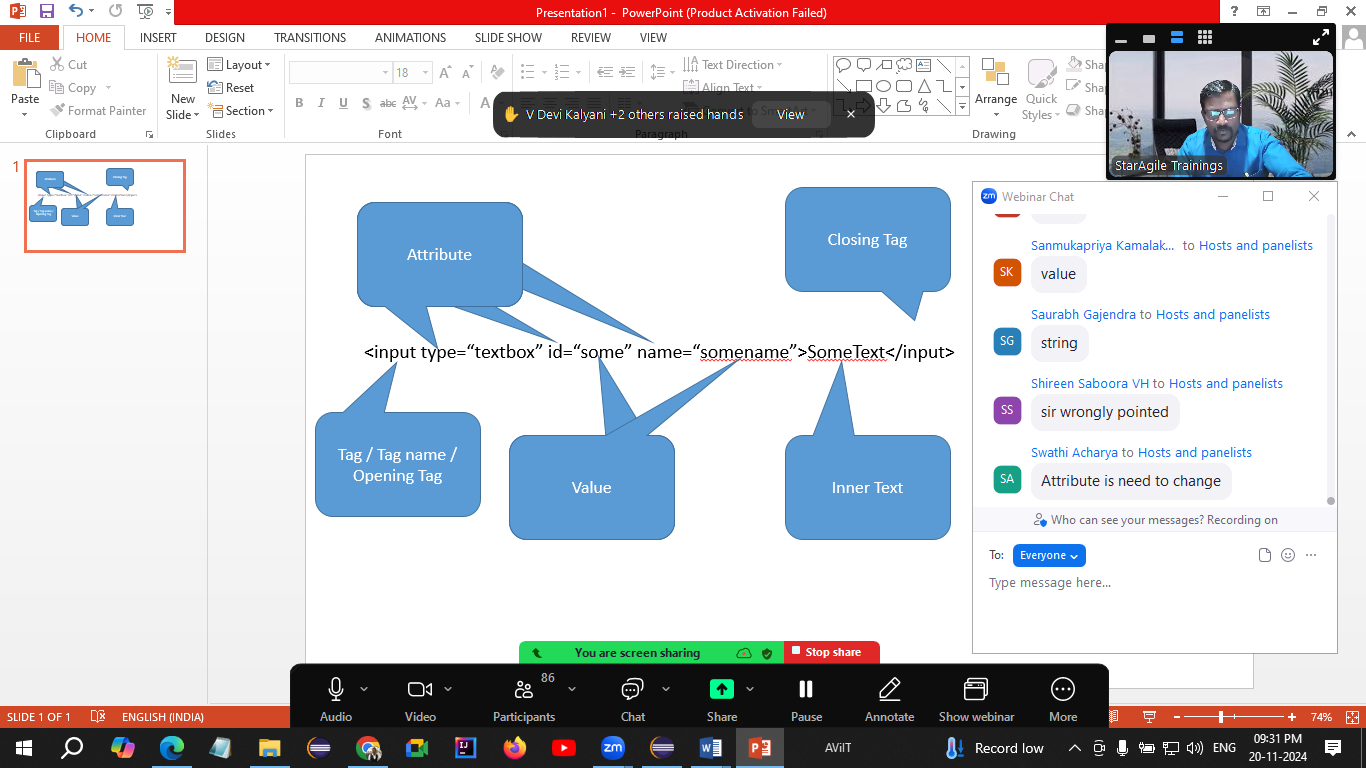
Project Structure of Java program in Eclipse

1. ~~Workspace (The folder where you are storing all your programs)~~
2. ~~Project~~
3. ~~Package~~
4. ~~Class~~

Selenium Methods

1. Launching the browser – Create object of WebDriver interface (This will launch the blank browser window)
2. get() – Used to launch the URL
3. driver.manage().window().maximize() – Will maximize the browser window.
4. close() – Will close the **current** browser window which is opened by WebDriver object.
5. getTitle() – This method will return the title of current web page. (String)
6. getCurrentUrl() – This method will return the URL of current web page (String)
7. getPageSource() – Returns the HTML source code of the page. (String)
8. findElement() – Used to read / find any control on the page. (WebElement). **Locates the first occurrence.**
9. findElements – used to read / find multiple controls on the page. (List<WebElement>)
10. getWindowHandles() – Returns ids of the browser windows those are opened by Selenium WebDriver. (Set<String>)

**If you want to perform any operation on any control, first you have to read that control in your script.**



WebElement:

* Every control on the page is treated as WebElement
  + Textbox
  + Radio button
  + Button
  + Checkbox
  + Dropdown list
* Interface in Selenium that holds any kind of control.
* Holds single control at a time.

**Methods of WebElement Interface**

1. sendKeys() – Used to enter some text in the textbox.
2. click() – Used to click on any control.
3. getText() – Used to return the text on control. (String)
4. isSelected() – Used to check whether the check box 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 the value of any specific attribute of the control. (String)

**Locators**

These are the way to uniquely identifying / find any control on the page.

1. Name
2. Id
3. ClassName – Class is the design added to the control. If the class value is separated by space, it means that there are multiple classes added to control. **Use any one class value from these.**
4. CssSelector
5. Xpath
6. LinkText
7. PartialLinkText
8. TagName
9. RelativeLocator

Common Exceptions

1. SessionNotCreatedException – The WebDriver version and Browser version are not matching
2. InvalidArgumentException – The URL is not in the correct format. (URL should be Absolute, starting with http)
3. NoSuchElementException – WebDriver is unable to locate or find this control
   1. The value of locator is wrong.
   2. The value of locator may be dynamic.
   3. Synchronization issue
   4. The control may be in iframe.
4. InvalidSelectorException – The format of Locator is wrong.
5. SessionTimeoutException – The page is not loaded by driver.get() within 30 seconds
6. ElementClickInterceptedException – Selenium is not able to click on the control due to the said control is hidden by some another control.
7. NoAlertPresentException – There is no alert you are trying to handle.
8. JavaScriptException – The parameters in js.executeScript are wrong.

CssSelector Locator

Cascading Style Sheet. (Used for designing the Web Page)

We can locate any control using any one or multiple attributes of a control.

Types

1. Single Attribute  
   Syntax:  
   tagName[attribute=”value”]  
   input[type="text"]
2. Multiple Attributes  
   Syntax:  
   tagName[attribute1=”value”][attribute2=”value”]
3. Special Characters
   1. ^ - Starts with  
      Syntax:  
      tagName[attribute^=”value”]
   2. $ - Ends With  
      Syntax:  
      tagName[attribute$=”value”]
   3. \* - Contains  
      Syntax:  
      tagName[attribute\*=”value”]
   4. # - id
   5. . – class name

Xpath Locator

(XML Path)

Pattankodoli Bus Stand 🡪 Take right turn 🡪 Water tank 🡪 Hupare Nagar Lane No 9 🡪 HNo 1128 (Ankush Vankore)

Types

1. Absolute Xpath  
   Stats with html
2. Relative Xpath  
   Starts with //  
   //tagName[@attribute=’value’]

**Handling drop down list**

If the control is having tag as “**select”** then only the control is treated as drop down list

**Select Class**

This is the class used to handle any drop down list or list box

This is the only way to handle drop down list or list box

Display selected country

Display total no of countries

Select United States (US)

Display selected country

Methods of Select class

1. getFirstSelectedOption() – used to return the element which is selected. (WebElement)  
   System.***out***.println("Selected Country: " + countries.getFirstSelectedOption().getText());
2. getOptions() – Return list of all the options from the drop down list. (List<WebElement>)  
   List<WebElement>countryList = countries.getOptions();
3. selectByVisibleText() – Select the option by using the text (Inner Text) on that option.  
   countries.selectByVisibleText("United States");
4. selectByValue() – Selects the option by using the attribute “value”. (String parameter)  
   countries.selectByValue("222");
5. selectByIndex() – Selects the option by using its zero based index. (int parameter)

cars.selectByIndex(1);

1. getAllSelectedOptions() – Returns the list of all selected options from List Box. (List<WebElement>)  
   List<WebElement> selectedCars = cars.getAllSelectedOptions();
2. isMultiple() – Checks whether the control is Dropdown list or List box. If it returns true means it is List Box otherwise it is Drop down list. (boolean)  
   **if**(cars.isMultiple() == **true**)
3. deSelectAll() – De select all the selected options from the list box  
   cars.deselectAll();

**Synchronization (Waits in Selenium)**

It is the process of adjusting the speed of tool with speed of application. (We cannot adjust the speed of application).

1. Thread.Sleep() – Will pause the execution of the script  
   Thread.*sleep*(5000);
   1. Takes the mandatory delay.
   2. It is applicable to single statement only.
2. ImplicitWait  
   driver.manage().timeouts().implicitlyWait(Duration.*ofSeconds*(10));
   1. It doesn’t takes mandatory delay.
   2. It is applicable throughout the script. (You are supposed to write this only once.)
3. ExplicitWait
   1. It is applicable to single statement only.
   2. It doesn’t takes mandatory delay.
   3. You can handle some conditions. (VisibilityOfElementLocated, ElementTobeClickable, AlertPresent)

wait.until(ExpectedConditions.*elementToBeClickable*(chkBox)).click();

1. FluentWait – Used where you want to handle synchronization along with Exceptions. This is next version of ExplicitWait
   1. It is applicable to single statement only.
   2. It doesn’t takes mandatory delay.
   3. You can handle some conditions. (VisibilityOfElementLocated, ElementTobeClickable, AlertPresent)
   4. You can handle Exceptions as well
   5. Uses following methods  
      w – withTimeout   
      i – ignoring   
      p – pollingEvery   
      u – until

wait.withTimeout(Duration.*ofSeconds*(10))

.ignoring(NoSuchElementException.**class**)

.pollingEvery(Duration.*ofMillis*(1))

.until(ExpectedConditions.*visibilityOfElementLocated*(By.*xpath*("//\*[@id=\"autoSuggestContainer\"]/div/div/div[1]/div/div[1]/ul/li[1]/div/text"))).click();

1. PageLoadTimeout – You can add some more delay to driver.get() to avoid SessionTimeoutException

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

**Handling Table**

1. Display all the headers
2. Display total number of rows
3. Display any row randomly

**Handling Alert (Javascript Alert)**

Alerts are not able to inspect.

You cannot perform any action on the web page if alert is displayed.

Alert is an interface in Selenium to handle such alerts.

Methods

1. switchTo().alert() – This method will switch to alert
2. getText() – Will return the text on Alert. (String)
3. accept() – Will click on Ok button of Alert.
4. dismiss() – Will click on Cancel button of Alert.
5. sendKeys() – Will enter the text on Prompt Box (Alert).

JavascriptExecutor – An interface to execute selenium script via Javascript.

**Robot Class** – This is the java class, via which you can perform some operations on OS windows (like Open file dialog box, save file dialog box etc.)

Mouse Actions

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

**Actions** Class is used to perform all above mouse actions.

Methods-

1. moveToElement() – Takes the mouse pointer to specific control. (Hover the mouse on control)
2. perform() – Performs the specific action on the control.
3. contextClick() – Right click on the control.
4. doubleClick() – Double click on the control.

TestNG (Test Next Generation)

It is a testing framework.

Framework – Set of rules and guidelines for automation testing. Which will make your automation testing much easier.

Advantages

* Collection of multiple tests
* Allows to set priorities for test cases
* Uses annotations
  + @Test
  + @BeforeTest
  + @AfterTest
  + @BeforeMethod
  + @AfterMethod
  + @DataProvider
  + @Parameters
  + @BeforeSuite
  + @AfterSuite
* Reusability
* Generates a report
  + Normal report
  + HTML report
* Supports for implementation of different frameworks
  + Linear framework
  + Modular Framework
  + Data Driven Framework
  + Keyword Driven Framework
  + Page Object Model
  + Hybrid Framework
* Allows to execute / skip single / multiple tests
* Allows creating the groups of test cases for execution.

Annotations

1. @Test – This is the ONLY method that is treated as test case  
   @Test (invocationCount = 5) will execute single test case for 5 times.
2. @BeforeTest – This is the method that will get executed ONLY ONCE BEFORE EXECUTING 1ST TEST CASE.
3. @AfterTest – This is the method that will get executed ONLY ONCE AFTER EXECUTING LAST TEST CASE
4. @BeforeMethod – This is the method that will get executed BEFORE EVERY TEST CASE.
5. @AfterMethod – This is the method that will get executed AFTER EVERY TEST CASE.
6. @DataProvider – This method sends the data to @Test method (Test Case)

Points to be noted about Configuration methods (Before & After)

1. They can appear anywhere in the code
2. They need not to be in pair.

Sequence of Methods

BeforeTest

BeforeMethod

Test 1

AfterMethod

BeforeMethod

Test 2

AfterMethod

BeforeMethod

Test 3

AfterMethod

BeforeMethod

Test 4

AfterMethod

AfterTest

**Data Driven Testing**

Executing single test case with multiple test data (data set)

* Array (2D Array)
* XML file
* Excel File

**Assertion in TestNG**

It is the way of marking any test case either as pass or as fail.

This can be used only in the case of Test Case only

**Modular Framework**

* Used to execute or skip single or multiple test cases.
* Allows to execute test cases via XML File.

While creating XML file please note…

1. All the XML tags are pre-defined
2. All the XML tags are case sensitive
3. You cannot alter the flow or sequence of XML Tags

<suite name = *"My Suite"*>

<test name = *"My Test"*>

<classes>

<class name = *"com.TestNGDemos.D05GoogleAssignment"*>

<methods>

<include name = *"businessLink"*></include>

</methods>

</class>

</classes>

</test>

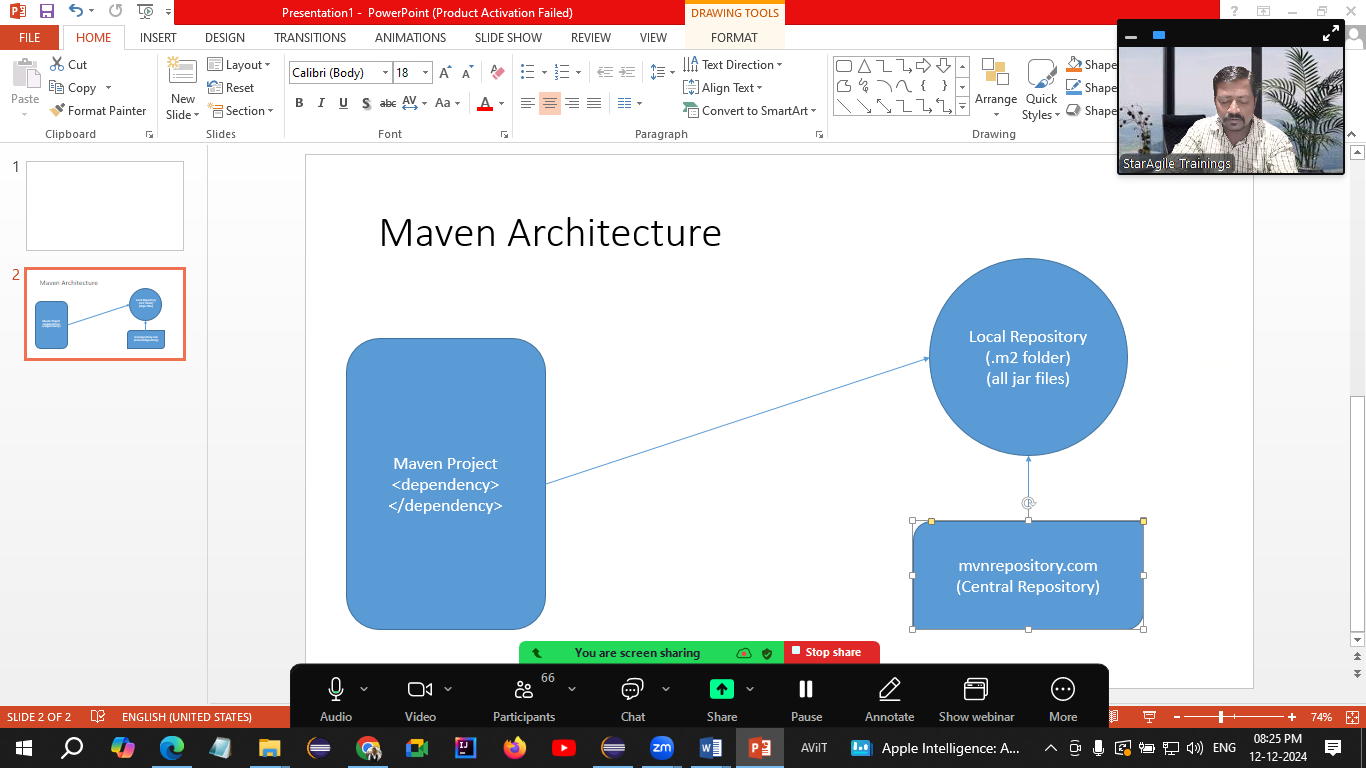
</suite>

Maven

* Apache Product
* Build management tool.
* Both developer and tester can use this tool.
* You can manage multiple projects.
* Makes configuration easier.
* Uses **pom.xml** file for configuration. (Project Object Model)
* Use dependencies. (This will replace your jar files)

Steps for creating new Maven project

1. File 🡪 New Maven Project
2. On New Maven Project window select 3rd checkbox
3. Next
4. Add the filter as **maven-archetype-quickstart**
5. Next
6. Give Artifact Id & Group ID (Both should be same)
7. Finish



Extent Report

This is the 3rd party tool to create test report.

<https://mvnrepository.com/artifact/com.aventstack/extentreports>

(Link for adding dependency)

TDD

BDD – Behavior Driven Development

BA, QA Team & Developer will gather the requirements from client

These requirements will be handed over to QA team and QA team will create **FEATURE FILE.** (First Component)

Using feature file

* Developer will implement the functionality.
* Manual Tester will create test cases.
* Automation tester will create the test script for the scenario.

After execution of feature file, you will get **STEP DEFINITION / GLUE CODE** (Second Component).

Step Definition contains automation script.

**RUNNER CLASS** (Third Component) is used to execute the code from step definition.

**Cucumber** is used to implement BDD Approach.

Steps for creating new Cucumber project

1. File 🡪 New Maven Project
2. On New Maven Project window select 3rd checkbox
3. Next
4. Add the filter as **io.cucumber**
5. Next
6. Give Artifact Id & Group ID (Both should be same)
7. Finish
8. Open src/test/java folder from package explorer
9. Delete the readymade package from this folder
10. Don’t delete anything from src/test/resources folder
11. Open pom.xml file
12. Delete the code from <properties> (line no 12) to </dependencies> (line no 59).
13. Open <https://github.com/cucumber/cucumber-java-skeleton/commit/d7249b50c570816eba27ce94557e1de7e9b0f97>
14. copy the code from <properties> (Line no 11) to </dependencies> (Line no 51) and paste in pom.xml file at line no 12
15. Delete following lines from pom.xml file
    1. <cucumber.version>6.8.2</cucumber.version>
    2. <maven.compiler.version>3.8.1</maven.compiler.version>
    3. <maven.surefire.version>2.22.2</maven.surefire.version>
16. Feature file
    1. Collection of Test Scenarios
    2. Having extension .feature
    3. Created using Gherkin language / syntax (Similar to English)
    4. Contains Keywords
       1. Feature: - Requirement
       2. Scenario: - Test Objective
       3. Given – Pre-Requisite
       4. When – Steps
       5. But / And – Combining multiple When statements
       6. Then – Expected Result
       7. Background: - Used when the same Given statements is used for multiple scenarios.
       8. Scenario Outline: - Used in the case of DDT
       9. Examples: - Used in the case of DDT
17. Step Definition
    1. Normal Java class (without main())
    2. Contains automation script
    3. Created using Cucumber annotation
18. Runner Class
    1. Normal Java class (without main())
    2. Created using JUnit annotations
    3. Used to execute automation script created in Step Definition.

Feature File

Requirement – Validate Google title

Steps

1. Open Google
2. Read the title
3. Title should be Google

Feature: Validate Google Title

Scenario: To validate title of Google home page

Given Open Google

When Read the title

Then Title should be Google

Requirement – Validate google search

Steps

1. Open Google
2. Enter valid text in search box
3. Hit Enter
4. Search result should display

Feature File

Feature: Validate Google Search

Scenario: To validate search functionality on Google

Given Open Google Application

When Enter valid text in search box

And Hit Enter

Then A valid search result should display

Tags in Cucumber

* These are special words those are added to Scenario
* Tags starts with @
* Tags are used to execution or skipping of any or multiple scenarios.
* Tags are always written in feature file.
* Execution is specified in runner class.

Hooks in Cucumber

* The methods those get executed before and after executing every scenario
* @Before – Executes before every scenario
* @After – Executed after every scenario