Ankush Vankore

Software Testing:

Process of checking the **correctness**, **completeness**, **security** and **quality** of developed software application.

Manual Testing:

Using your hand – eye – brain co-ordination.

1. Entering data in text box
2. Clicking on some links
3. Clicking on button
4. Selecting single option from multiple options (List box, drop down list, check box, radio button)

What is mean by Automation?

Automation is nothing but performing all above action with some automation testing tool

Java Basics for revision

1. Basics (Variables, operators,)
2. Conditions (if, if-else, switch case, nested if else)
3. Loops (for & enhanced for loop)
4. Wiring and calling methods
5. Creating object of class and calling the methods via class
6. Creating and using static members and methods
7. Collections
   1. List
   2. Set
   3. Map
8. Exception Handling
   1. throws keyword

Automation Testing

Advantages

* Save time
* It saves money
* It saves human errors
* Reduces human efforts
* Reusable
* Full coverage
* More productivity
* More accuracy
* Efficiency
* Consistency
* Get the report
* Cost saving

Need of automation:

* Faster execution
* Saves time
* Full coverage
* Accuracy
* Increases performance
* Good quality of testing
* Reporting
* Reduces work load
* More innovative
* Compability testing

When to Automate:

* Stable build
* Regression testing
* Repeated test scenario
* Large number of test cases
* Large amount of data to be tested
* Cross browser testing
* Performance testing
* Parallel testing

Types of automation testing

* Unit Testing
* API Testing (Application Programming Interface)
* GUI Testing
* Mobile Testing
* Performance Testing
* Security Testing

Tools

* JUnit / NUnit
* Selenium
* Postman
* Cucumber
* Tosca
* Appium
* REST Assured
* Load Runner
* HP Web Scrab
* Maven
* QTP
* Winium

Process of Automation:

1. Planning
2. Selection of Tool
   * Technology
   * Cost of tool
   * Human Resources availability
   * Support availability
   * Market presence
3. Create test script
4. Execute the script
5. Generate Report
6. Maintance

**Selenium**

**Suite / Bundle of Test Automation Tools used to test *Web / Browser Based Applications*** (Web Sites)

**Components of Selenium**

* Selenium IDE – Record and playback
* Selenium Grid – Parallel Execution
* Selenium RC (Remote Control) – Deprecated
* Selenium WebDriver – In replacement of RC

Installing Selenium IDE

Open selenium.dev 🡪 Downloads tab 🡪 scroll to Selenium IDE 🡪 Choose your browser and click on that browser name 🡪 Click on Add Extention

**Selenium WebDriver**

* A test automation tool to test web based applications (Web Sites)
* WebDriver is an **interface** in java.
* It is an API

**Pre – Requisite**

1. Windows OS
   1. Minimum 10
2. Java
   1. Minimum version: 11
3. Editor
   1. Eclipse / Intellij
4. Any updated browser
   1. Chrome / Firefox / Edge / Opera / Safari
5. Latest Selenium Jar file (API)

**Configuration Part**

1. Create 2 folders (mostly except C: )
   1. YourName\_SeleniumDemos
   2. WebDriver Jar Files
2. Open <https://www/selenium.dev>
3. Click on **Downloads**
4. Click on Latest stable version [4.24.0](https://github.com/SeleniumHQ/selenium/releases/download/selenium-4.24.0/selenium-server-4.24.0.jar) (Version may vary) **(DON’T OPEN THIS FILE)**
5. Store this file in WebDriver Jar Files
6. Open Eclipse
   1. Select the workspace (The folder where you are going to store your project) – Select the folder created in 1st step YourName\_SeleniumDemos
7. Click on Launch
8. File 🡪 New 🡪 Java Project
9. Give Name to project
10. Select Java version as Java-SE-11
11. Uncheck **create module-info.java file** checkbox
12. Create a new package inside this project
13. Create a class inside this package
14. Right click on project (in Package Explorer)
15. Click on Build Path
16. Click on Configure Build Path
17. Click on Libraries Tab
18. Click on Class Path
19. Click on Add External Jars…
20. Browse and select the File downloaded (Selenium-Server-xxxx.jar)
21. Click on Open
22. Click on Apply & Close

Selenium WebDriver

1. Launching the browser: Create the object of WebDriver interface. It will launch blank browser page.
2. get(): Launch the specified url
3. driver.manage().window().maximize(): Maximize the browser window
4. close() : Close the current browser window
5. getTitle() : Read the title of page (String)
6. getCurrentUrl() : Read the URL of page (String)
7. getPageSource() :
8. findElement() : Reads the single WebElement / Control on the page using any one locator. **Always locates first occurrence.** (WebElement)
9. findElements() : Used to read multiple controls on the page. (List<WebElement>)
10. getWindowHandles() : Returns the ids of all the browser windows those are opened by Selenium. (Set<String>)
11. quit() : Will close all the browser windows those are opened by Selenium.

**WebElement**

* Interface
* This refers any control on the page
* Methods
  + sendKeys() : Used to enter some text in the textbox (Will append the text)  
    sendKeys(Keys.Control + “A”) : will select all the text from Text box
  + click() : used to click on any control
  + getText() : Used to read the text on the control (String)
  + isSelected() : Check whether the control (Checkbox / Radio button) is selected or not. (boolean – true / false)
  + isDisplayed() : Check whether the control is visible or not (boolean)
  + isEnabled() : Check whether the control is enabled or disabled (boolean)
  + getAttribute() : Returns the value of any attribute. (String)

Common Exception in Selenium WebDriver

1. InvalidArgumentException: Your url is wrong (URL should always start by https means it should be ABSOLUTE URL)
2. SessionNotCreatedException: Your browser and WebDriver version are mismatching.
3. NoSuchElementException: Selenium is unable to locate this control because of some reason.
   1. Value of locator is wrong
   2. Value of locator may be dynamic. (Every time it is changing.)
   3. Synchronization issue
   4. May possible the control is inside the **iframe**
4. InvalidSelectorException: The value of locator is in wrong format.
5. ElementNotInteractableException: You are not able to interact with this control.
6. TimeoutException : The site is not getting load in 30 seconds
7. ElementClickInterceptedException : The specified control is hidden by some another control. (Use JavascriptedExecutor for further operations)

**Locators**

These are the ways to find / locate any WebElement on the page.

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

<input type="text" class="inputtext \_55r1 \_6luy" name="email" id="email" data-testid="royal\_email" placeholder="Email address or phone number" autofocus="1" aria-label="Email address or phone number">

TagName

Attribute

Value

CssSelector:

Can be used to read the control by using any one of the attribute of control.

1. Using single attribute  
   tagName[attribute=”value”]  
   input[type="text"]
2. Using multiple attributes  
   tagName[attribute1=”value”] [attribute2=”value”]  
   input[placeholder="Password"][aria-label="Password"]
3. Using special characters
   1. ^ - Starts with  
      tagName[attribute^=”value”]
   2. $ - Ends with  
      tagName[attribute$=”value”]
   3. \* - Contains   
      tagName[attribute\*=”value”]
   4. # - id  
      tagName#id value
   5. . – className  
      tagName.class – specify single class only

XPath

XML Path

Pattankodoli Bus Stand > Take a Right turn > Hupare Nagar > Lane no 9 > House No 1128

Types

1. Absolute Xpath  
   starts with html tag  
   html/<innter tag>/inner tag
2. Relative Xpath
   1. To get the xpath of any control via parent control  
      //tagName[@attribute=”value”]/tagName
   2. To get the xpath of control directly  
      //tagName[@Attribute=”value”]
3. Following-Sibling – Will give you the controls next to the current control  
   //\*[@id="leftcontainer"]/table/tbody/tr[12]/td[1]/following-sibling::td  
   //\*[@id="leftcontainer"]/table/tbody/tr[12]/td[1]/following-sibling::\*
4. Preceding-sibling – will give all the controls before the specified control in the same hierarchy   
   //\*[@id="leftcontainer"]/table/tbody/tr[12]/td[6]/preceding-sibling::td  
   //\*[@id="leftcontainer"]/table/tbody/tr[12]/td[6]/preceding-sibling::\*
5. Contains Text  
   //span[text()="16"]  
   //span[contains(text(), "16")]  
   //span[text()="Invalid Username/Password"]  
   //input[contains(@id, "But")]  
   //\*[contains(text(), "16")]

**Handling Dropdown List**

**If the control is having <select> tag then only it is treated as drop down list**

**Select is the class used to handle dropdown list / list box**

**Methods of Select class**

1. getFirstSelectedOption() : Return the selected option from the list. (WebElement)
2. getOptions() : Return list of all the options from list. (List<WebElement>)
3. selectByVisibleText() : Selects the option by using the text on that option.
4. selectByValue() : Selects the option by using its value attribute
5. selectByIndex() : Selects the option by using its index (Zero based)
6. getAllSelectedOptions() : Returns list of all selected options (List<WebElement>)
7. deselectAll() : Deselects all the selected options

Difference between List box & Dropdown List

1. Size of control
2. List box allows to select multiple options
3. In list box multiple options are visible where as in dropdown list only one option is visible

**Synchronization (Waits in Selenium)**

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

1. Thread.sleep() : is used to pause the execution of script for specified time duration. (Milliseconds)
   1. Applicable to single statement only
   2. It takes mandatory delay
2. ImplicitWait
   1. It don’t take mandatory delay
   2. It is applicable to the entire script
3. ExplicitWait (WebDriverWait)
   1. Applicable to single statement only
   2. It don’t take mandatory delay
4. FluentWait – Next version of ExplicitWait
   1. Applicable to single statement only
   2. It don’t take mandatory delay
   3. We can handle Exception as well  
      w – withTimeout   
      i – ignoring   
      p – pollingEvery   
      u – until
5. PageLoadTimeout

**Handling Table**

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

**Handling Alert (Javascript Alert)**

If the pop-up is not inspect able then only it is an alert.

**Alert** interface is used to handle Alert in selenium.

Methods:

1. switchTo().alert() – Takes you on Alert
2. getText() – Returns the text on Alert (String)
3. accept() – Will click on Ok button on Alert
4. dismiss() – Will click on Cancel button on Alert.
5. sendKeys() – Used to enter some text on Alert (Prompt Box / Input Box)

**Mouse Actions**

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

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

**TestNG**

TestNG is a testing framework.

Framework: Set of rules/methods/classes for execution of test cases. Via which your automation will more easily.

Advantages of TestNG

* Combine multiple tests together
* Set priorities for test cases
* Use annotations
  + @Test
  + @BeforeTest
  + @AfterTest
  + @BeforeMethod
  + @AfterMethod
  + @DataProvider
  + @Parameters
* Implement data driven testing
* Get Report
  + Normal report
  + HTML Report
* Parameterization
* Execute / Skip single / multiple tests

**Annotations**

1. @Test: This method is treated as test case
2. @BeforeTest: This method will get executed **only once before executing 1st test case.**
3. @AfterTest: This method will get executed **only once after executing last test case.**
4. @BeforeMethod: This method will get executed **before executing every test case.**
5. @AfterMethod: This method will get executed **after executing every test case.**
6. @DataProvider: This method sends the data to test case in the form of 2D Array.
7. @Parameters: This annotation will read the parameters from XML file.

**Points to be noted about @Before & @After**

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

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 multiple times with multiple data set

* Using Array (@DataProvider)
* Using XML file
* Using Excel File

**Modular Framework**

* Executing or skipping single / multiple test cases
* Executing test cases via XML file

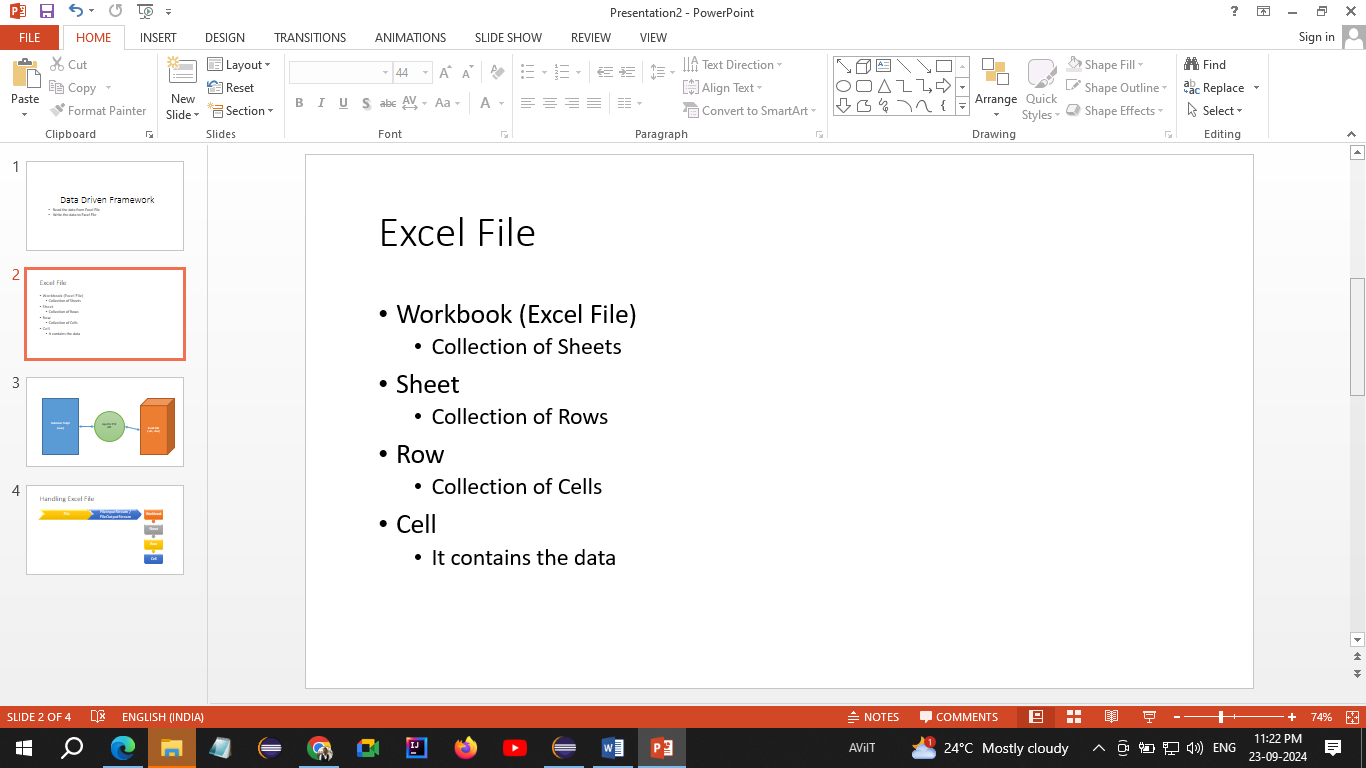
**Points to be noted while creating XML file for modular framework**

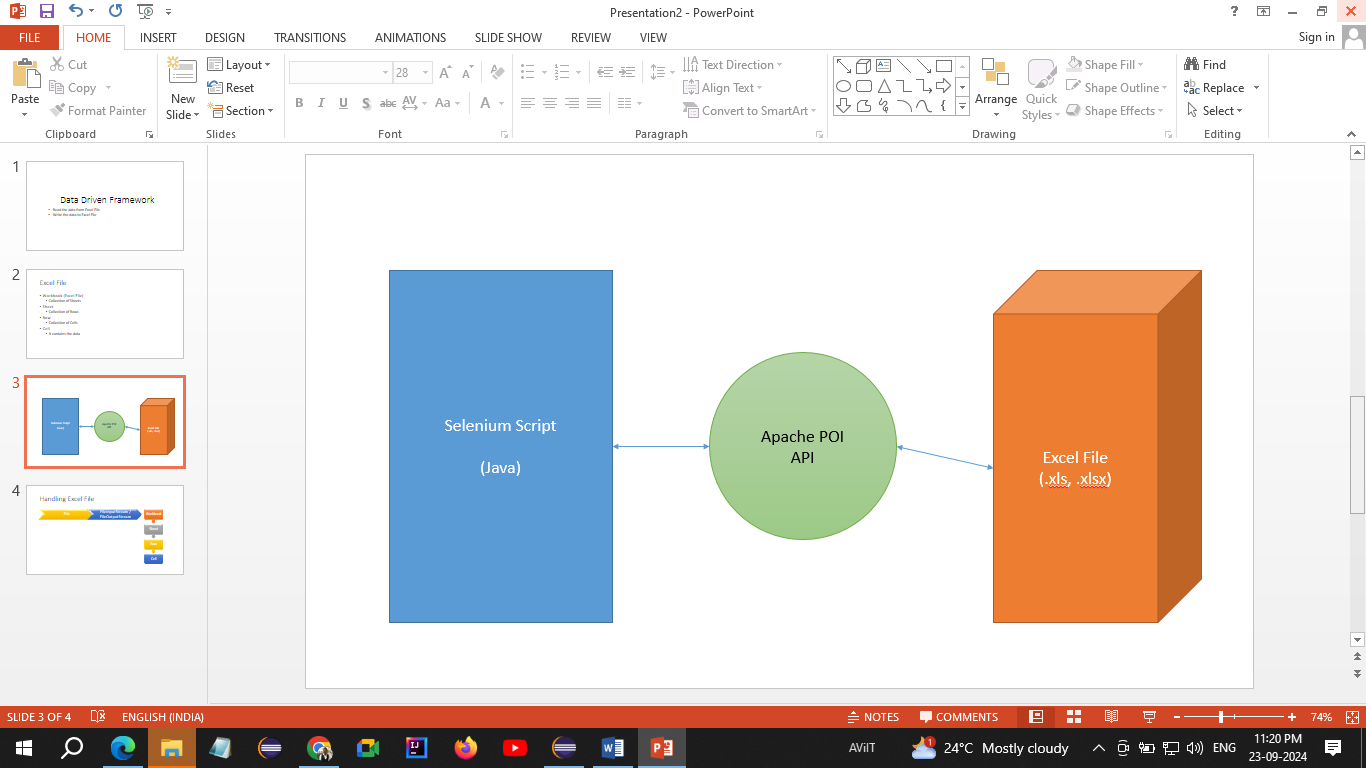
* All the tags are pre-defined
* All tags are case sensitive
* You cannot alter sequence of any tag

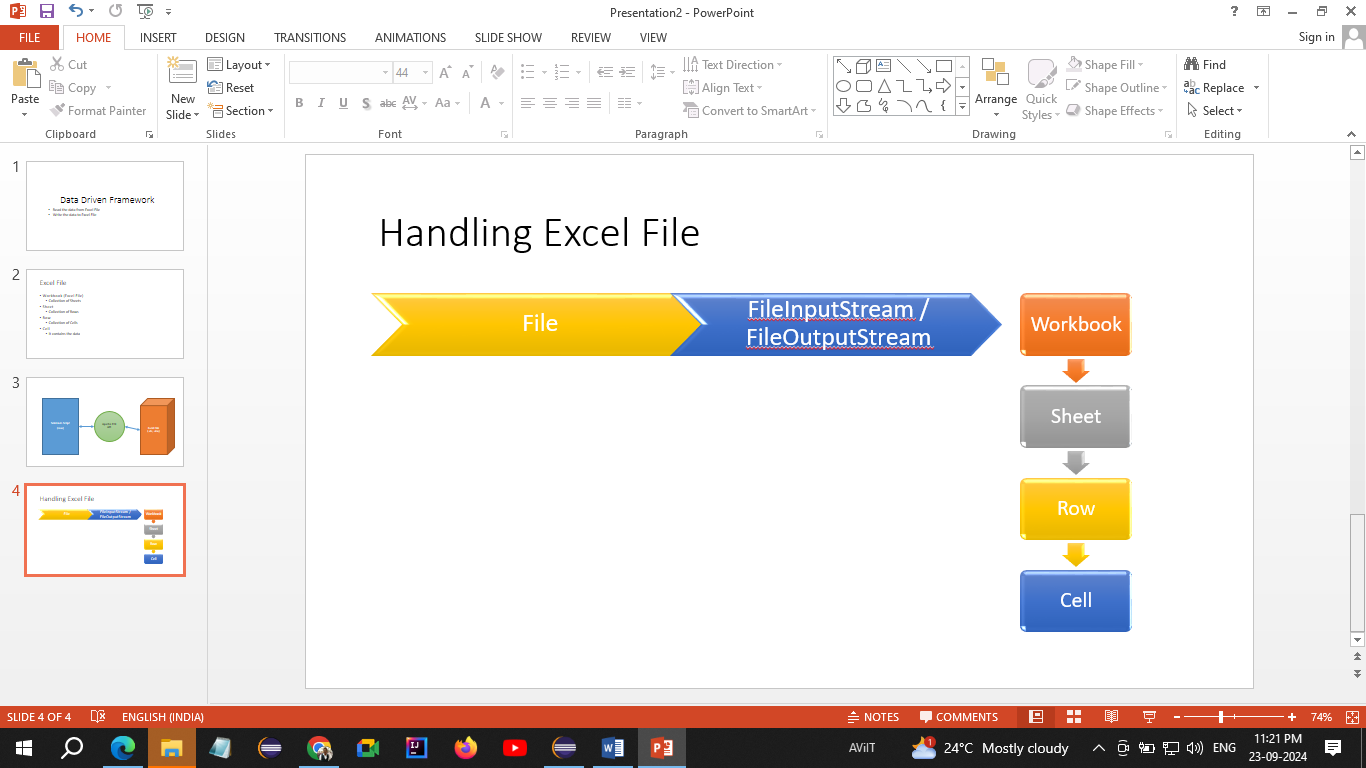
**Keyword Driven Framework**: Reading the data from .properties file / Excel file

**Page Object Model (POM):** Creating multiple methods for implementing the logic of script and calling those methods whenever required.

Data Driven Framework







Types of Testing Tools

* Test Management Tools
  + Jira
  + TestLink
  + QC (Quality Center)
  + Test Rail
* Defect Tracking / Management Tools
  + JIRA
  + Bugzilla
  + Asana
  + QC
  + Bugasura
* Automation Testing Tools
  + Functional Testing Tools
    - Selenium
    - Tosca
    - QTP
  + Non Functional Testing Tools
    - JMeter
    - Load Runner
    - HP Web Inspect
  + API Testing Tools
    - Postman
    - REST Assured

Maven:

* Build management tool.
* It is Apache product.
* Both developer and tester can use Maven.
* You can manage multiple projects using Maven at the same time.
* Uses dependencies.
* Uses pom.xml for configuration of project. (pom 🡪 Project Object Model)

Cucumber

TDD – Test Driven Development

BDD – Behavior Driven Development

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

**Feature File**

* Having .feature as extension.
* Created by QA team
* Gherkin Syntax / Language (Similar to English)
* Uses some keywords
* Collection of Scenarios (Functionality to be tested)
* This file is shared with developer, so that developer can generate the code / functionality
* Same file is shared with manual tester, so that tester will create test cases
* Same file is shared with automation tester, so that automation tester will create automation script.

After executing this feature file I will get

**Step Definition / Glue Code**

* Class that contains the logic / code for automation
* Contains JUnit annotations

**Test Runner (Runner class)**

* It is a normal class
* It is used to execute the automation script
* Using JUnit annotations

**Keywords in Feature file.**

1. Feature: - The requirement
2. Scenario: - Test case Objective
3. Given – The pre-requisite
4. When – The steps to be performed
5. And – Used to combine multiple steps (When statement)
6. Then – The expected result
7. Background: - When some common steps (Epically Given) needs to be performed.
8. Examples: - Used in the case of Data Driven Testing
9. Scenario Outline: Used in the case of Data Driven Testing

Creating Cucumber Project

File 🡪 New 🡪 Maven Project 🡪 Select 3rd Checkbox 🡪 Next

Enter io.cucumber in filter (Select the filter) 🡪 Next

Enter Group id & artifact id 🡪 Finish

Folder structure

* src/test/java – This folder contains Step definition and runner class
* src/test/resources – This folder contains the Feature files
* target – Contains the reports
* pom.xml – xml configuration for your project

Important setting in pom.xml file

1. Select the code from <properties> to </dependencies > (Line no 12 to line no 59)
2. Delete this code
3. Open <https://github.com/cucumber/cucumber-java-skeleton/commit/d7249b50c570816eba27ce94557e1de7e9b0f97>
4. Select the code from <properties> till </dependencies> (Line no 11 to 41)
5. Copy this code and paste in pom.xml file
6. Save the file
7. Change the version of java at <java.version>11</java.version>
8. Delete <cucumber.version>6.8.2</cucumber.version>
9. Delete <maven.compiler.version>3.8.1</maven.compiler.version> <maven.surefire.version>2.22.2</maven.surefire.version>

**Creating Feature file**

Inside src/test/resources folder 🡪 right click 🡪 file

**Google Title**

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

Feature: Check Google Title

Scenario: To validate title of Google

Given Open Google

When Read the title of page

Then Title should be Google

**Google Search**

1. Open Google
2. Enter some text in the search box
3. Hit enter
4. Valid search result should display

Feature: Google Search Functionality

Scenario: To validate google search functionality

Given Open Google in browser

When I Enter valid search text

And I hit enter

Then Valid search should display

Hooks in Cucumber

These are the two methods those get executed before first scenario and after last scenario.

@Before

@After

**Point to be noted: Hooks are never the part of .feature file.**

You can either create a separate package for hooks or even you can write the hook in the same step definition class.