**CUCUMBER BDD FRAMEWORK – BASICS AND ADVANCED SYLLABUS**

**CUCUMBER FRAMEWORK – BASICS – DAY ‘1’**

1. [Java JDK Setup](about:blank)
2. [Apache Maven Setup](about:blank)
3. Eclipse Setup
4. [Eclipse IDE Setup](about:blank)
5. [Cucumber Add-on for Eclipse](about:blank)
6. Chrome Browser & Chrome Driver Setup
7. Creating Sample Maven Project from Eclipse
8. CUCUMBER BASICS - Introduction
9. What is a Feature File? [What is Gherkin? What exactly is a Feature File? - How are they connected?](about:blank)
10. [Creating Our First Feature File](about:blank)
11. Step File Creation
12. Runner Class Creation
13. [CUCUMBER ADVANCED](about:blank) - Inspecting Advanced Gherkin Keywords!
14. Gherkin Advanced Keywords
15. Simulating Multiple Test Scenarios
16. Adding Multiple Scenarios - Introduction
17. Creating Multiple Feature Files
18. Cucumber Parameterization (Data Driven Testing)
19. Data Tables (Structuring Our Test Data)
20. Scenario Outlines – Introduction
21. Scenario Outlines – Code Example & Explanation
22. Scenario Outlines – Adding Hooks - Practical Explanation
23. Scenario Outlines - Adding Multiple Test Scenarios
24. Scenario Outlines – Enhancing Test Cases with Assertions
25. Hooks - Introduction & Explanation

**CUCUMBER FRAMEWORK – ADVANCED – DAY ‘2’**

1. Cucumber Options
2. Options - Introduction & Explanation
3. Options - Code Example & Explanation
4. Tags - Introduction & Explanation
5. Tags - Code Example & Explanation
6. Connecting Tags & Hooks
7. Tagged Hooks – Introduction & Explanation
8. Creating Multiple Runner Classes
9. Generating Advanced Reports
10. Advanced Reporting – Introduction & Explanation
11. Batch File Creation (Multiple Ways to Execute our Test Framework)
12. Continuous Integration (Jenkins Setup and Generating Cucumber Reports)
13. Installing & Running Jenkins
14. Jenkins Implementation 1-of-4 - Installing Essential Add-ons
15. Jenkins Implementation 2-of-4 - Connecting Java JDK & Apache Maven to Jenkins
16. Jenkins Implementation 3-of-4 - Project Creation - Execute Projects When We Want
17. Jenkins Implementation 4-of-4 - Generating Advanced Cucumber Reports & More!
18. Extent (Advanced) Reporting Walk-through
19. Dependency and ExtentConfig.xml setup
20. How can we attach unique images to our reports?
21. Developing advanced methods with image capturing capabilities
22. Looking into Extents Report methods and extending upon our existing methods
23. Connecting Extents ReportConfig.xml to our existing framework
24. Attaching test images to Extent Reports
25. Looking into pre-constructed Extent Report methods

**Selenium and Cucumber BDD Troubleshooting steps**

1. [How to rerun failed test cases of cucumber-jvm in jenkins](about:blank)

**Solution** - Use the rerun plugin to generate a list of scenarios that failed:

java cucumber.api.cli.Main --plugin rerun:rerun.txt features

This will write the location of each failing scenario to a text file called rerun.txt (you can call it whatever you want). Then you can use the output file as an input to your next run of Cucumber to specify which scenarios should be executed:

java cucumber.api.cli.Main < rerun.txt

1. [Cucumber java throws single undefined step when other steps are working fine](about:blank).

**Solution** - Turned out that the issue was in my Steps definition class.

Change the annotation from

@Then ("^Validate in total, how much is (.+) owed by everyone else. They are owed £(.d)$")

to

@Then ("^Validate in total, how much is (.\*) owed by everyone else. They are owed £(.\*)$")

1. **As a user, I would like to exclude a class ( Test Runner ) when I am running mvn clean test**

**Sample XML suite file**

1. ?xml version="1.0" encoding="UTF-8"?>
2. <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
3. <suite name="Automated UI Tests">
4. <test name = "Cucumber Test" verbose="2">
5. <classes>
6. <class name="com.testrunners.SampleTest"/>
7. <class name="com.testrunners.Sample2Test"/>
8. <class name="com.testrunners.Sample3Test"/>
9. </classes>
10. </test>
11. </suite>

**Sample Test Runner:**

1. package com.testrunners;
2. import cucumber.api.CucumberOptions;
3. import cucumber.api.junit.Cucumber;
4. import cucumber.api.testng.AbstractTestNGCucumberTests;
5. import org.junit.runner.RunWith;
6. @RunWith(Cucumber.class)
7. @CucumberOptions(
8. features = {
9. "src/test/java/com/features/",
10. },
11. glue = {
12. "com.stepdefinitions"
13. },
14. monochrome = true,
15. tags = {
16. "@smoke"
17. },
18. plugin = {"pretty",
19. "html:target/cucumber/sample",
20. "json:target/cucumber-report/sample/cucumber.json",
21. "json:target/sample/cucumber.json",
22. "com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter: target/sample/report.html"}
23. )
24. public class SampleTest extends AbstractTestNGCucumberTests {
25. }

**Solution - From Maven Failsafe plugin:** You can use excludes property in POM.xml. Please see t**he configuration** to do it. Reference from Maven failsafe official doc plugin.

1 <project>

2 [...]

3 <build>

4 <plugins>

5 <plugin>

6 <groupId>org.apache.maven.plugins</groupId>

7 <artifactId>maven-failsafe-plugin</artifactId>

8 <version>3.0.0-M4</version>

9 <configuration>

10 <excludes>

11 <exclude>\*\*/CircleIT.java</exclude>

12 <exclude>\*\*/SquareIT.java</exclude>

13 </excludes>

14 </configuration>

15 </plugin>

16 </plugins>

17 </build>

18 [...]

19 </project>

1. [Reports are not generated when the build is failed in Maven Cucumber Reports](about:blank)

Reports are generating successfully when the build is successful but when there are any failed cases which cause build failure, reports are not generated

Add the following configuration to the sure fire plugin. It will not stop the maven execution after the failure. Then it will generate the report.

<testFailureIgnore>true</testFailureIgnore>

as given below with your existing configuration.

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.20</version>

<configuration>

<testFailureIgnore>true</testFailureIgnore>

</configuration>

</plugin>

1. [How to take screenshot with Selenium WebDriver](about:blank)

Yes, it is possible. The following example is in Java:

WebDriver driver = new FirefoxDriver();

driver.get("http://www.google.com/");

File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

// Now you can do whatever you need to do with it, for example copy somewhere

FileUtils.copyFile(scrFile, new File("c:\\tmp\\screenshot.png"));

1. [Debugging “Element is not clickable at point” error](about:blank)

This is caused by following 3 types:

**1.The element is not visible to click.**

Use **Actions** or **JavascriptExecutor** for making it to click.

By Actions:

WebElement element = driver.findElement(By("element\_path"));

Actions actions = new Actions(driver);

actions.moveToElement(element).click().perform();

By JavascriptExecutor:

JavascriptExecutor jse = (JavascriptExecutor)driver;

jse.executeScript("scroll(250, 0)"); // if the element is on top.

jse.executeScript("scroll(0, 250)"); // if the element is on bottom.

or

JavascriptExecutor jse = (JavascriptExecutor)driver;

jse.executeScript("arguments[0].scrollIntoView()", Webelement);

Then click on the element.

**2.The page is getting refreshed before it is clicking the element.**

For this, make the page to wait for few seconds.

**3. The element is clickable but there is a spinner/overlay on top of it**

The below code will wait until the overlay disppears

By loadingImage = By.id("loading image ID");

WebDriverWait wait = new WebDriverWait(driver, timeOutInSeconds);

wait.until(ExpectedConditions.invisibilityOfElementLocated(loadingImage));

Then click on the element.

1. [Wait for page load in Selenium](about:blank)

If you set the implicit wait of the driver, then call the findElement method on an element you expect to be on the loaded page, the WebDriver will poll for that element until it finds the element or reaches the time out value.

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

1. [How to verify an XPath expression in Chrome Developers tool or Firefox's Firebug?](about:blank)

**By using chrome or Opera**

**without any plugins, without writing any single XPath syntax character**

1. right click the required element, then "inspect"
2. right click on highlighted element tag, choose copy>Copy Xpath
3. [How to run Selenium WebDriver test cases in Chrome?](about:blank)

You need to download the executable driver from: [ChromeDriver Download](about:blank)

Then all you need to do is use the following before creating the driver object (already shown in the correct order):

System.setProperty("webdriver.chrome.driver", "/path/to/chromedriver");

WebDriver driver = new ChromeDriver();

1. [Test if element is present using Selenium WebDriver?](about:blank)

Use findElements instead of findElement.

findElements will return an empty list if no matching elements are found instead of an exception.

To check that an element is present, you could try this

Boolean isPresent = driver.findElements(By.yourLocator).size() > 0

**For more Selenium and Cucumber BDD trouble shooting steps refer the following sites:**

1. Selenium Web Driver Highly Voted Troubleshooting steps - [https://stackoverflow.com/questions/tagged/selenium-webdriver?tab=Votes](about:blank)
2. Newest cucumber Troubleshooting steps - [https://stackoverflow.com/questions/tagged/cucumber](about:blank)