Selenium Day 4 - Classroom Session – Trainer's Handbook

Topics : Assertions + TestNG + BDD (Cucumber, Gherkin)

Session Agenda

1. **Understanding Assertions in Selenium**
2. **Introduction to TestNG Framework**
3. **Annotations, Parallel Execution, and Reporting in TestNG**
4. **Introduction to BDD (Cucumber & Gherkin)**
5. **Writing Test Cases in Cucumber Using Gherkin Syntax**
6. **Hands-on Exercises (During the Session)**
7. **Best Practices & Interview Questions**
8. **Post-Classroom Assignment (2 Hours)**
9. **Expected Outcomes by End of Day 4**

Session Details

1. Understanding Assertions in Selenium

What are Assertions?

- Assertions help verify that the expected result of a test case matches the actual result.
- If the assertion fails, the test stops execution at that point.
- Types of Assertions:
    - **Hard Assertions (Stops execution on failure).**

- **Soft Assertions (Continues execution even if it fails).**

A. Hard Assertions in Selenium (JUnit/TestNG)

```java
import org.testng.Assert;

public class HardAssertionExample {
    public static void main(String[] args) {
        String actualTitle = "Google";
        String expectedTitle = "Google";

        Assert.assertEquals(actualTitle, expectedTitle, "Title does not match!");
        System.out.println("This will be executed only if assertion passes.");
    }
}
```

B. Soft Assertions in Selenium (TestNG SoftAssert Class)

```java
import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;

public class SoftAssertionExample {
    @Test
    public void testSoftAssertions() {
        SoftAssert softAssert = new SoftAssert();

        softAssert.assertEquals("Hello", "Hello", "First assertion passed!");
        softAssert.assertEquals(5, 10, "Second assertion failed!");

        System.out.println("This will be executed even if an assertion fails.");

        softAssert.assertAll();  // Collects all failed assertions at the end.
    }
}
```

---

2. Introduction to TestNG Framework

Why Use TestNG?

- Supports parallel test execution.
- Generates detailed HTML reports.
- Provides annotations to control test execution flow.
- Handles dependencies between test cases.

A. TestNG Annotations Overview

Annotation	Description
@Test	Marks a method as a test case
@BeforeMethod	Runs before each test method
@AfterMethod	Runs after each test method
@BeforeClass	Runs once before all test cases in a class
@AfterClass	Runs once after all test cases in a class
@BeforeSuite	Runs before all tests in the suite
@AfterSuite	Runs after all tests in the suite

B. Writing a Basic TestNG Script

```java
import org.testng.annotations.Test;

public class TestNGExample {
    @Test
    public void firstTest() {
        System.out.println("Executing First Test");
    }

    @Test
    public void secondTest() {
        System.out.println("Executing Second Test");
    }
}
```

C. Running Tests in Parallel (Parallel Execution in TestNG XML)

- Create a testng.xml file and configure parallel execution:

```xml
<suite name="ParallelSuite" parallel="tests" thread-count="2">
    <test name="Test1">
        <classes>
            <class name="com.example.TestNGExample"/>
        </classes>
    </test>

    <test name="Test2">
        <classes>
            <class name="com.example.AnotherTestClass"/>
        </classes>
    </test>
</suite>
```

---

3. Introduction to BDD (Cucumber & Gherkin)

What is BDD?

- **BDD (Behavior-Driven Development)** is an approach that bridges the gap between business and technical teams by using **plain English syntax** for test cases.
- **Cucumber** is a tool that implements BDD using **Gherkin syntax**.

A. Gherkin Syntax Basics

Keyword	Description
Feature	Describes the feature under test
Scenario	Defines a test case
Given	Sets up preconditions
When	Defines actions performed by the user
Then	Specifies expected outcomes

B. Sample Gherkin Test Case

```
Feature: Login Functionality
  Scenario: Successful Login with Valid Credentials
    Given User is on the login page
    When User enters valid username and password
    Then User should be redirected to the homepage
```

---

4. Writing Test Cases in Cucumber Using Gherkin Syntax

A. Step Definitions in Java (Cucumber Step Definition Class)

```java
import io.cucumber.java.en.*;

public class LoginSteps {
    @Given("User is on the login page")
    public void userOnLoginPage() {
        System.out.println("User navigates to login page");
    }

    @When("User enters valid username and password")
    public void enterCredentials() {
        System.out.println("User enters credentials");
    }

    @Then("User should be redirected to the homepage")
    public void verifyHomePage() {
        System.out.println("User is on homepage");
    }
}
```

B. Running Cucumber Tests Using TestNG

- Create a **Runner Class** to execute Gherkin test cases:

```java
import io.cucumber.testng.AbstractTestNGCucumberTests;
import io.cucumber.testng.CucumberOptions;

@CucumberOptions(
    features = "src/test/resources/features",
    glue = "stepDefinitions"
)
public class TestRunner extends AbstractTestNGCucumberTests {
}
```

Hands-on Exercises (During the Session)

✅ **Exercise 1:** Implement Hard and Soft Assertions in a test case. (Solution)
✅ **Exercise 2:** Run TestNG tests in parallel using an XML file. (Solution)
✅ **Exercise 3:** Create a BDD scenario using Gherkin and automate it with Selenium. (Solution)
✅ **Exercise 4:** Implement dependency management in TestNG. (Solution)

Best Practices & Interview Questions

Best Practices:

✔ Always use **Hard Assertions** for critical validations and **Soft Assertions** when you want the script to continue even after failure.

- ✔ In TestNG, use @BeforeSuite and @AfterSuite for setup and cleanup tasks.
- ✔ Keep **Cucumber step definitions reusable** for maintainability.
- ✔ Organize BDD test cases **logically in feature files**.

Interview Questions:

- ❓ What is the difference between **Hard Assertions and Soft Assertions**?
- ❓ How do you achieve **parallel execution in TestNG**?
- ❓ What is the difference between **JUnit and TestNG**?
- ❓ Explain **the purpose of Gherkin keywords** like Given, When, Then.
- ❓ How do you integrate **Cucumber with TestNG**?
- ❓ What are **hooks** in Cucumber? How do you use them?

Post-Classroom Assignment (2 Hours)

Task 1: Write a TestNG script with **BeforeClass, BeforeMethod, AfterMethod, and AfterClass annotations**. (Solution)
Task 2: Create a BDD test for **Search functionality on an e-commerce website** using Cucumber. (Solution)
Task 3: Implement **Soft Assertions** in an existing Selenium test case. (Solution)
Task 4: Run **Cucumber test cases in parallel** using TestNG. (Solution)

Expected Outcomes by End of Day 4

- ✔ Mastery of **Assertions in Selenium** for validation.
- ✔ Strong understanding of **TestNG annotations, parallel execution, and reporting**.
- ✔ Ability to write and execute **Cucumber BDD test cases using Gherkin syntax**.
- ✔ Confidence to answer **client interview questions** on these topics.
- ✔ Hands-on experience with **real-world automation scenarios**.

Solutions

Exercise 1: Implement Hard and Soft Assertions in a test case

Hard assertions will stop the test execution if the assertion fails, while soft assertions will continue the test execution even if the assertion fails.

Here's an example using TestNG:

```
import org.testng.Assert;
import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;

public class AssertionsExample {

    @Test
    public void testHardAssertion() {
```

```
        System.out.println("Hard Assertion Test Started");
        Assert.assertEquals("Hello", "Hello");
        System.out.println("This line will be executed because the
assertion passed.");
        Assert.assertEquals("Hello", "World");
        System.out.println("This line will not be executed because the
assertion failed.");
    }

    @Test
    public void testSoftAssertion() {
        SoftAssert softAssert = new SoftAssert();
        System.out.println("Soft Assertion Test Started");
        softAssert.assertEquals("Hello", "Hello");
        System.out.println("This line will be executed because the
assertion passed.");
        softAssert.assertEquals("Hello", "World");
        System.out.println("This line will be executed even though the
assertion failed.");
        softAssert.assertAll(); // This will collect all the assertion
results and mark the test as failed if any assertion failed.
    }
}
```

Exercise 2: Run TestNG tests in parallel using an XML file

To run TestNG tests in parallel, you can configure the `testng.xml` file.

Here's an example of a `testng.xml` file:

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="ParallelSuite" parallel="tests" thread-count="2">
    <test name="Test1">
        <classes>
            <class name="com.example.TestClass1"/>
        </classes>
    </test>
    <test name="Test2">
        <classes>
            <class name="com.example.TestClass2"/>
        </classes>
    </test>
</suite>
```

Exercise 3: Create a BDD scenario using Gherkin and automate it with Selenium

To create a BDD scenario using Gherkin and automate it with Selenium, you can use Cucumber.

Here's an example:

Feature File (`login.feature`):

```
Feature: Login functionality
```

```
  Scenario: Successful login with valid credentials
    Given the user is on the login page
    When the user enters valid credentials
    And the user clicks the login button
    Then the user should be redirected to the homepage
```

Step Definitions (`LoginSteps.java`):

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import io.cucumber.java.en.*;

public class LoginSteps {
    WebDriver driver;

    @Given("the user is on the login page")
    public void the_user_is_on_the_login_page() {
        System.setProperty("webdriver.chrome.driver",
"path/to/chromedriver");
        driver = new ChromeDriver();
        = driver.findElement(By.id("loginButton"));
        loginButton.click();
    }

    @Then("the user should be redirected to the homepage")
    public void the_user_should_be_redirected_to_the_homepage() {
        String expectedUrl = "https://example.com/home";
        String actualUrl = driver.getCurrentUrl();
        Assert.assertEquals(expectedUrl, actualUrl);
        driver.quit();
    }
}
```

Exercise 4: Implement dependency management in TestNG

To implement dependency management in TestNG, you can use the `dependsOnMethods` attribute.

Here's an example:

```
import org.testng.annotations.Test;

public class DependencyExample {

    @Test
    public void testLogin() {
        System.out.println("Login test executed");
        // Add login test logic here
    }

    @Test(dependsOnMethods = {"testLogin"})
    public void testAddToCart() {
        System.out.println("Add to Cart test executed");
        // Add add to cart test logic here
```

```
    }

    @Test(dependsOnMethods = {"testAddToCart"})
    public void testCheckout() {
        System.out.println("Checkout test executed");
        // Add checkout test logic here
    }
}
```

Task 1: Write a TestNG script with BeforeClass, BeforeMethod, AfterMethod, and AfterClass annotations

Here's an example of a TestNG script using `@BeforeClass`, `@BeforeMethod`, `@AfterMethod`, and `@AfterClass` annotations:

```
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.AfterClass;
import org.testng.annotations.Test;

public class TestNGAnnotationsExample {

    @BeforeClass
    public void setUpClass() {
        System.out.println("BeforeClass: Set up class-level resources.");
    }

    @BeforeMethod
    public void setUpMethod() {
        System.out.println("BeforeMethod: Set up method-level resources.");
    }

    @Test
    public void testMethod1() {
        System.out.println("Test Method 1: Execute test logic.");
    }

    @Test
    public void testMethod2() {
        System.out.println("Test Method 2: Execute test logic.");
    }

    @AfterMethod
    public void tearDownMethod() {
        System.out.println("AfterMethod: Clean up method-level
resources.");
    }

    @AfterClass
    public void tearDownClass() {
        System.out.println("AfterClass: Clean up class-level resources.");
    }
}
```

Task 2: Create a BDD test for Search functionality on an e-commerce website using Cucumber

To create a BDD test for search functionality using Cucumber, you need to define a feature file and step definitions.

Feature File (`search.feature`):

```
Feature: Search functionality

  Scenario: Successful search with valid keyword
    Given the user is on the homepage
    When the user enters a valid keyword in the search bar
    And the user clicks the search button
    Then the search results should be displayed
```

Step Definitions (`SearchSteps.java`):

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import io.cucumber.java.en.*;

public class SearchSteps {
    WebDriver driver;

    @Given("the user is on the homepage")
    public void the_user_is_on_the_homepage() {
        System.setProperty("webdriver.chrome.driver",
"path/to/chromedriver");
        driver = new ChromeDriver();
        driver.get("https://example-ecommerce.com");
    }

    @When("the user enters a valid keyword in the search bar")
    public void the_user_enters_a_valid_keyword_in_the_search_bar() {
        WebElement searchBar = driver.findElement(By.id("searchBar"));
        searchBar.sendKeys("laptop");
    }

    @When("the user clicks the search button")
    public void the_user_clicks_the_search_button() {
        WebElement searchButton =
driver.findElement(By.id("searchButton"));
        searchButton.click();
    }

    @Then("the search results should be displayed")
    public void the_search_results_should_be_displayed() {
        WebElement searchResults =
driver.findElement(By.id("searchResults"));
        assert searchResults.isDisplayed();
        driver.quit();
    }
}
```

Task 3: Implement Soft Assertions in an existing Selenium test case

To implement Soft Assertions in an existing Selenium test case, you can use the `SoftAssert` class from TestNG.

Here's an example:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;

public class SoftAssertionsExample {

    @Test
    public void testSoftAssertions() {
        System.setProperty("webdriver.chrome.driver",
"path/to/chromedriver");
        WebDriver driver = new ChromeDriver();
        driver.get("https://example.com");

        SoftAssert softAssert = new SoftAssert();

        WebElement element1 = driver.findElement(By.id("element1"));
        softAssert.assertTrue(element1.isDisplayed(), "Element 1 is not
displayed.");

        WebElement element2 = driver.findElement(By.id("element2"));
        softAssert.assertEquals(element2.getText(), "Expected Text",
"Element 2 text does not match.");

        WebElement element3 = driver.findElement(By.id("element3"));
        softAssert.assertTrue(element3.isEnabled(), "Element 3 is not
enabled.");

        softAssert.assertAll(); // This will collect all the assertion
results and mark the test as failed if any assertion failed.

        driver.quit();
    }
}
```

Task 4: Run Cucumber test cases in parallel using TestNG

To run Cucumber test cases in parallel using TestNG, you need to configure the `testng.xml` file and use the `@CucumberOptions` annotation.

TestNG XML File (`testng.xml`):

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="ParallelSuite" parallel="tests" thread-count="2">
    <test name="CucumberTest1">
        <classes>
```

```
            <class name="com.example.RunCucumberTest"/>
        </classes>
    </test>
    <test name="CucumberTest2">
        <classes>
            <class name="com.example.RunCucumberTest"/>
        </classes>
    </test>
</suite>
```

Runner Class (`RunCucumberTest.java`):

```
import io.cucumber.testng.AbstractTestNGCucumberTests;
import io.cucumber.testng.CucumberOptions;

@CucumberOptions(
    features = "src/test/resources/features",
    glue = "com.example.steps",
    plugin = {"pretty", "html:target/cucumber-reports.html"}
)
public class RunCucumberTest extends AbstractTestNGCucumberTests {
}
```