Cucumber Test Automation using Selenium & JUnit

1. Installations and Configurations – Cucumber Maven Project

Prerequisites

- **Java JDK 8+** installed
- **Maven** installed
- **Eclipse/IntelliJ IDEA** (or any IDE)
- **Browser & WebDriver** (e.g., Chrome & ChromeDriver)

Create Maven Project

```
mvn archetype:generate -DgroupId=com.example -
DartifactId=CucumberSeleniumJUnit \
-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

Add Dependencies in `pom.xml`

```
<dependencies>
    <!-- Selenium -->
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>4.21.0</version>
    </dependency>

    <!-- Cucumber JVM -->
    <dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-java</artifactId>
        <version>7.18.0</version>
    </dependency>

    <!-- Cucumber JUnit -->
    <dependency>
        <groupId>io.cucumber</groupId>
        <artifactId>cucumber-junit</artifactId>
        <version>7.18.0</version>
        <scope>test</scope>
    </dependency>

    <!-- JUnit -->
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.13.2</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

2. Simple Scenarios

Feature File – `login.feature`

```
Feature: Login functionality

  Scenario: Valid login with correct credentials
    Given User is on Login Page
    When User enters username "admin" and password "password123"
    Then User should be navigated to Home Page
```

Step Definition – `LoginSteps.java`

```
import io.cucumber.java.en.*;
import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;
import static org.junit.Assert.*;

public class LoginSteps {
    WebDriver driver;

    @Given("User is on Login Page")
    public void user_is_on_login_page() {
        driver = new ChromeDriver();
        driver.get("https://example.com/login");
    }

    @When("User enters username {string} and password {string}")
    public void user_enters_username_and_password(String username, String
password) {
        driver.findElement(By.id("username")).sendKeys(username);
        driver.findElement(By.id("password")).sendKeys(password);
        driver.findElement(By.id("loginBtn")).click();
    }

    @Then("User should be navigated to Home Page")
    public void user_should_be_navigated_to_home_page() {
        assertTrue(driver.getTitle().contains("Home"));
        driver.quit();
    }
}
```

3. Scenario Outlines (Data-Driven with Examples)

Feature File – `login_outline.feature`

```
Feature: Login functionality with multiple datasets

  Scenario Outline: Valid and Invalid login attempts
    Given User is on Login Page
    When User enters username "<username>" and password "<password>"
    Then Login should be "<status>"

  Examples:
    | username | password     | status  |
    | admin    | password123  | success |
    | user1    | wrongpass    | failure |
```

Step Definition – Reusing previous step

```
@Then("Login should be {string}")
public void login_should_be(String status) {
    if (status.equals("success")) {
        assertTrue(driver.getTitle().contains("Home"));
    } else {
        assertTrue(driver.getPageSource().contains("Invalid credentials"));
    }
    driver.quit();
}
```

---

4. Data-Driven using Data Tables

Feature File – `datatable.feature`

```
Feature: Login with multiple users

  Scenario: Enter multiple user credentials
    Given User is on Login Page
    When User logs in with credentials
      | username | password   |
      | admin    | admin123  |
      | user1    | pass123   |
    Then Verify login attempts
```

Step Definition – `DataTableSteps.java`

```
import io.cucumber.datatable.DataTable;

@When("User logs in with credentials")
public void user_logs_in_with_credentials(DataTable dataTable) {
    List<Map<String, String>> users = dataTable.asMaps();
    for (Map<String, String> user : users) {
        driver.findElement(By.id("username")).sendKeys(user.get("username"));
        driver.findElement(By.id("password")).sendKeys(user.get("password"));
        driver.findElement(By.id("loginBtn")).click();
    }
}
```

---

5. Data-Driven using Lists

Feature File – `lists.feature`

```
Feature: Search functionality

  Scenario: Search for multiple products
    Given User is on Search Page
    When User searches for products
      | Laptop   |
      | Mobile   |
      | Headset  |
    Then Results should be displayed
```

Step Definition

```
@When("User searches for products")
public void user_searches_for_products(List<String> items) {
    for (String item : items) {
        driver.findElement(By.id("searchBox")).sendKeys(item);
        driver.findElement(By.id("searchBtn")).click();
        System.out.println("Searched for: " + item);
    }
}
```

---

6. Dependency Injection (Using PicoContainer)

Add dependency:

```
<dependency>
  <groupId>io.cucumber</groupId>
  <artifactId>cucumber-picocontainer</artifactId>
  <version>7.18.0</version>
</dependency>
```

Example:

```
public class SharedDriver {
    private final WebDriver driver;

    public SharedDriver() {
        driver = new ChromeDriver();
    }

    public WebDriver getDriver() {
        return driver;
    }
}
public class LoginSteps {
    private final SharedDriver shared;

    public LoginSteps(SharedDriver shared) {
        this.shared = shared;
    }

    @Given("User is on Login Page")
    public void user_is_on_login_page() {
        shared.getDriver().get("https://example.com/login");
    }
}
```

---

7. Parallel Execution

Configure in `pom.xml`

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>3.2.5</version>
    <configuration>
        <parallel>methods</parallel>
        <threadCount>4</threadCount>
    </configuration>
</plugin>
```

JUnit Runner – `RunCucumberTest.java`

```
import org.junit.runner.RunWith;
import io.cucumber.junit.Cucumber;

@RunWith(Cucumber.class)
@io.cucumber.junit.CucumberOptions(
    features = "src/test/resources/features",
    glue = "stepdefinitions",
    plugin = {"pretty","html:target/cucumber-report.html"},
    monochrome = true
)
public class RunCucumberTest {}
```

---

8. Feature Level Parallelism

Cucumber allows **splitting feature files** across multiple runners.

- Create multiple runner classes (`RunLoginTest`, `RunSearchTest`)
- Configure Surefire to run in parallel with `parallel=classes`

```
<configuration>
    <parallel>classes</parallel>
    <threadCount>2</threadCount>
</configuration>
```

---

9. Reporting – Generating Reports

Pretty Format (HTML Report)

Add to Runner:

```
@CucumberOptions(
  plugin = {"pretty","html:target/cucumber-
report.html","json:target/cucumber.json"}
)
```

Using Extent Reports

Add dependency:

```
<dependency>
  <groupId>tech.grasshopper</groupId>
  <artifactId>extentreports-cucumber7-adapter</artifactId>
  <version>1.13.0</version>
</dependency>
```

Add plugin in Runner:

```
plugin = {"pretty","html:target/cucumber-html-report",
          "json:target/cucumber.json",

"com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:"}
```

Reports generated under `target/`.