What is Cucumber? How to use it?

Cucumber testing is an automation tool that supports the Behavior Driven Development (BDD) approach.

Cucumber allows you to write test scenarios in a understandable language called Gherkin. Gherkin enables you to define test scenarios with keywords such as Feature, Scenario, Given, When, Then, And, But.

Cucumber can automatically execute these scenarios and report the results also can be used with various programming languages like Ruby, Python, Java, Javascript, etc.

Feature File:

To use Cucumber testing, you first need to define the feature you want to test as a “Feature:”

> 💡 For example, something like “Login to the Amazon site and search for a product.”

```
Feature: Login to the Amazon site and search for a product
```

Next, you create scenarios to test this feature.

> 💡 For example, scenarios like “User logs in with email and password and searches for a product.”

```
Scenario: User logs in with email and password and searches for a product
```

Then, you need to describe each step of these scenarios using keywords like Given, When, Then, And, But.

💡 For instance, “Given Amazon page is loaded, When User enters email and password and logs in, Then Home page is displayed, And User searches for a product, Then Search results are displayed.”

```
Given Amazon page is loaded
When User enters email and password and logs in
Then Home page is displayed
And User searches for a product
Then Search results are displayed
```

This way, you can easily write and read your test scenarios.

```
Feature: Login to the Amazon site and search for a product

  Scenario: User logs in with email and password and searches for a product
    Given Amazon page is loaded
    When User enters email and password and logs in
    Then Home page is displayed
    And User searches for a product
    Then Search results are displayed
```

Java Class Implementing Steps:

Next, you need to create a Java class that performs the steps in this feature file. You can create this class under the “Steps” package.

💡 For example, you can create a Java class like the one below:

```java
package Steps;

import Base.BaseStep;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import org.junit.Assert;
import org.openqa.selenium.By;
import org.openqa.selenium.Keys;

public class AmazonStep extends BaseStep {

    @Given("Amazon page is loaded")
    public void amazonPageIsLoaded() {
        driver.get("https://www.amazon.com/");
        driver.manage().window().maximize();
    }

    @When("User enters email and password and logs in")
    public void userEntersEmailAndPasswordAndLogsIn() {
        click(By.xpath("//div[@class='account-nav-item user-login-containe
r']"));
        sendKeys(By.id("login-email"), "test@test.com");
        sendKeys(By.id("login-password-input"), "123456");
        click(By.xpath("//button[@type='submit']"));
    }

    @Then("Home page is displayed")
    public void homePageIsDisplayed() {
        Assert.assertTrue(isElementDisplayed(By.xpath("//div[@class='account-nav
-item user-login-container']")));
    }

    @And("User searches for a product")
    public void userSearchesForAProduct() {
        sendKeys(By.className("search-box"), "iphone 15" + Keys.ENTER);
    }

    @Then("Search results are displayed")
    public void searchResultsAreDisplayed() {
        Assert.assertTrue(isElementDisplayed(By.xpath("//div[@class='prdct-cntnr
-wrppr']")));
    }
}
```

In this Java class, we've labeled the steps from the feature file using annotations (@Given, @When, @And, @Then,) matching the names. Additionally, we extended this class from the BaseStep class created under the "Base" package.

This class contains methods using Selenium WebDriver to perform operations like opening a web page, clicking elements, entering data, and checking elements. We can use these methods by calling them from the BaseStep class.

Test Runner Class:

Finally, you need to create a Runner class to run your test scenarios. You can create this class under the "Runner" package.

For example, you can create a Runner class like the one below:

```
package Runner;

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

@RunWith(Cucumber.class)
@CucumberOptions(
        features = "src/test/java/Features",
        glue = {"Steps"}
)
public class TestRunner {
}
```

In this Runner class, we specified where the feature files and Java classes are located. Additionally, using the Cucumber library, we configured the necessary settings to run our test scenario.

The advantages of using Cucumber testing:

You can write your test scenarios in a clear and natural language, Gherkin. This allows you to share and get approval for your test scenarios from people who may not

know how to code.

You can separate your test scenarios and code. This means that when you need to change your test scenarios, you don’t have to change your code. Similarly, when you need to change your code, you don’t have to change your test scenarios.

You can reuse your test scenarios and code. This eliminates the need to repeatedly write similar test scenarios or code.

I hope this helps you better understand and use Cucumber. See you!