Cucumber and Test NG

TestNG (Test Next Generation): Test Framework

> 💡 It is an open-source test framework for the Java programming language. It aims to simplify and strengthen the test process.

Inspired by the JUnit test framework and added many new features.

Cucumber: Test Framework

> 💡 It is a test automation tool that supports the BDD (Behavior Driven Development) approach. This approach enables collaboration in writing, understanding, and verifying functional requirements in software projects.

Cucumber also allows you to create Step Definitions. Step Definitions are code snippets that link the steps in Gherkin scenarios to Java code.

Differences and Similarities between TestNG and Cucumber

Differences:

TestNG:

- Uses Java annotations* to define test scenarios.
- Allows you to control the order* and dependencies of tests.
- It has robust documentation and a large user community.

Cucumber:

- It uses the Behavior Driven Development (BDD) approach.
- Defines test scenarios with Gherkin text files written in natural language.
- Matches step definitions with Java code.
- Facilitates execution and reporting of tests.
- Can align test scenarios with business requirements and enable collaboration among business analysts, developers, and testers.

Similarities:

- They are open-source frameworks for test automation.
- They support unit, integration, and end-to-end tests.
- Enable reusability of test scenarios.
- Present test results in report formats.
- Provide logging capabilities for reporting and debugging.
- Have features for running tests in parallel or in batches.
- Can be integrated with popular CI/CD (Continuous Integration/Continuous Deployment) tools (Jenkins, Travis CI, GitLab CI, etc.).
- Both are tightly integrated into the Java ecosystem and offer more features and functionality. Java is recommended to fully leverage their potential.

  Example:

  - When using TestNG with Java, you can easily control the order* and dependencies of your tests.

```
@Test(priority = 1)
public void test1() {
  // ...
}

@Test(priority = 2)
```

```
public void test2() {
  // ...
}

@Test(dependsOnMethods = {"test1", "test2"})
public void test3() {
  // ...
}
```

```
from unittest import TestCase, sortTestMethods

class TestExample(TestCase):

    test_methods = sortTestMethods([
        'test_example3',
        'test_example2',
        'test_example1',
    ])

    def test_example1(self):
        # ...

    def test_example2(self):
        # ...

    def test_example3(self):
        # ...
```

- When using Cucumber with Java, you can easily match step definitions with Java code.

```
@Given("^I am on the login page$")
public void onLoginPage() {
  // ...
}

@When("^I enter my username and password$")
public void enterCredentials() {
  // ...
}

@Then("^I should be logged in$")
public void loggedIn() {
  // ...
}
```

```
require 'cucumber'

Given /^I am on the login page$/ do
  # ...
end

When /^I enter my username and password$/ do
  # ...
end

Then /^I should be logged in$/ do
  # ...
```

```
end
```

Advantages of Using Java:

1. **Widespread Usage:** Java is a popular programming language with a large developer community, so using Cucumber framework with Java allows you to benefit from a wide community and resource pool.
2. **Integration Ease:** Java, being widely used in the industry, can be easily integrated with other components written in Java.
3. **Performance:** Java is known as a performance-oriented language. Therefore, it can provide better performance when processing large and complex test scenarios.
4. **Development Tools:** Java has a rich development ecosystem. Integrated development environments (IDEs) such as Eclipse, IntelliJ IDEA provide useful tools for writing and managing Java-based Cucumber tests.
    a. "Cucumber for Java" plugin.

TestNG may be a better choice in these cases:

- If you need more types of tests.
- If you want more control and flexibility.
- If you need to integrate it into an existing Java project.

Cucumber may be a better choice in these cases:

- If you want to use the Behavior Driven Development (BDD) approach.
- If you want to write test scenarios in natural language.
- If you want to facilitate communication with stakeholders.

Differences and Similarities between TestNG and Cucumber:

Feature	TestNG	Cucumber

Test Approach	Traditional	Behavior Driven Development (BDD)
Test Scenario Language	Java annotations*	Gherkin
Java Integration	Required	Required
Test Types	Unit, integration, end-to-end	Unit, integration, end-to-end
Reporting	Detailed reports	Simple reports

Consider these factors when deciding which framework to choose.

Additional Information and Details

▼ Annotations

TestNG Annotations:

- **@Test:** Defines a test method.
- **@BeforeTest:** Executed once before a test class.
- **@AfterTest:** Executed once after a test class.
- **@BeforeMethod:** Executed before each test method.
- **@AfterMethod:** Executed after each test method.
- **@BeforeClass:** Executed as a static method before a test class.
- **@AfterClass:** Executed as a static method after a test class.
- **@Test(groups = {"group1", "group2"}):** Assigns a test method to one or more groups.
- **@BeforeGroups:** Executed before tests in specified groups start.
- **@AfterGroups:** Executed after tests in specified groups finish.
- **@Parameters:** Used to pass parameters to a test method.
- **@DataProvider:** Used to provide data to a test method.
- **@ExpectedExceptions:** Specifies which exceptions a test method should throw.

- **@Ignore:** Used to ignore a test method or class.
- **@Timeout:** Prevents a test method from exceeding a specified time.
- **@DependsOnMethods:** Specifies that a test method depends on other test methods.

▼ Sequencing

The `priority` attribute determines the order in which tests will run.
The
`dependsOnMethods` attribute determines that a test will run after other tests have completed.

▼ CI/CD (Continuous Integration/Continuous Deployment)

CI/CD stands for **Continuous Integration and Continuous Deployment**. It encompasses a set of practices and principles used to automate the integration, testing, and delivery of code changes continuously throughout the software development process.

CI/CD Tools:

- **Continuous Integration (CI):** Enables automated merging, testing, and error detection with every code change.
- **Continuous Deployment (CD):** Facilitates the automatic deployment of developed code to production or testing environments.

Benefits of CI/CD Tools:

- Accelerates the software development process.
- Improves software quality.
- Reduces deployment risks.
- Enhances team productivity.

Popular CI/CD Tools:

- **Jenkins:** An open-source, popular CI/CD tool.
- **GitHub Actions:** A CI/CD tool integrated with GitHub.
- **Azure DevOps:** A CI/CD platform provided by Microsoft.

- **GitLab CI:** A CI/CD tool integrated with GitLab.

> 💡 **Example:**
>
> A software development team is working on a new feature for a web application. The team is manually performing the following steps with each code change:
>
> - Submits code changes to a repository.
> - Manually merges code changes.
> - Manually runs tests.
> - Manually fixes errors.
> - Manually deploys code to production.
>
> **Using a CI/CD Tool:**
>
> The team can automate this process using a CI/CD tool like Jenkins. The tool will automatically perform the following steps with each code change:
>
> - Retrieves code changes from a repository.
> - Automatically merges code changes.
> - Runs unit tests and integration tests automatically.
> - Automatically reports errors.
> - Deploys code to production automatically if tests pass.

▼ Dependencies

```
htmlCopy code
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-junit</artifactId>
    <version>7.14.0</version>
```

```
    <scope>test</scope>
</dependency>
```

Specifies a dependency within the test scope (i.e., to be applied only to tests).

It will enable its use with the JUnit test runner.

```
htmlCopy code
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>7.14.0</version>
</dependency>
```

To enable Cucumber-Java integration.

```
htmlCopy code
        <dependency>
            <groupId>org.testng</groupId>
            <artifactId>testng</artifactId>
            <version>7.8.0</version>
        </dependency>
```

To use the TestNG framework.