TestNG in Test Automation

What is TestNG and its usage?

What is TestNG?

- TestNG (Test Next Generation) is a testing framework for Java designed to simplify various levels of automated testing.

Different types of testing that can be automated using TestNG :-

- Unit
- Integration
- End to End
- Regression
- Smoke
- Sanity
- Data Driven & many more..

Now, How to write a test using TestNG

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

public class CalculatorTest {

    @Test
    public void testAdd() {
        // Arrange
        Calculator calculator = new Calculator();
        int num1 = 5;
        int num2 = 10;
        int expectedResult = 15;

        // Act
        int result = calculator.add(num1, num2);

        // Assert
        Assert.assertEquals(result, expectedResult, "Addition result is incorrect");
    }
}
```

// Example using JAVA

What are Assertions?

As can be seen in previous slide, assert was used to validate the result.

But why?

Reason to use Assertions - to verify the expected behavior or outcomes of the test. To compare actual result matches with expected result or not!

Now how can we use assertions?

```
Assert.assertEquals(result, expectedResult, "Addition result is incorrect");
```

Different types of assertions:-

- assertEquals(expected, actual, message): Compares whether the expected and actual values are equal.
- assertTrue(condition, message): Checks whether the given condition is true.
- assertFalse(condition, message): Checks whether the given condition is false.
- assertNotNull(object, message): Verifies that the object is not null.

What is an effective way to write Tests?

The test scenario to be automated should be breakdown into 3 key components:-

- Arrange
- Act
- Assert

Now how to use this technique Automation?

- Including arranging the test scenario
- Performing the action
- Asserting the expected outcome

How to write Tests using AAA

```
// Function to perform addition

public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }
}

// Test

    @Test
    public void testAdd() {
        // Arrange
        Calculator calculator = new Calculator();
        int num1 = 5;
        int num2 = 10;
        int expectedResult = 15;
        // Act
        int result = calculator.add(num1, num2);
        // Assert
        Assert.assertEquals(result, expectedResult, "Addition result is incorrect");
    }
}
```

What are different annotations in TestNG

Before what are different annotations, What does annotations mean?

Annotations are a form of metadata that provides additional information about code to the compiler. They do not change the action of a compiled program but can affect how the program is treated or processed.

Helps in enabling developers to provide essential information and metadata that can be utilized by tools, frameworks, and the Java runtime environment to enhance code behavior, maintainability, and readability.

Now what are different annotations, can be found in next slide —-->

@Test:
Purpose: Marks a method as a test method.
Usage: Denotes the method that performs the actual test logic.

@BeforeMethod:
Purpose: Executes before each test method.
Usage: Use for setting up preconditions or initializing common objects before each test method.

@AfterMethod:
Purpose: Executes after each test method.
Usage: Use for cleaning up resources or performing post-test actions after each test method.

@BeforeClass:
Purpose: Executes before the first test method in the test class.
Usage: Typically used for setting up one-time configurations or resources needed for the entire test class.

@AfterClass:
Purpose: Executes after all test methods in the test class have run.
Usage: Used for cleaning up resources or performing post-test actions for the entire test class.

@BeforeSuite:
Purpose: Executes before all tests in a suite.
Usage: Used for global setup or configurations required for the entire test suite.

@AfterSuite:
Purpose: Executes after all tests in a suite have run.
Usage: Used for global cleanup or actions required after the execution of the entire test suite.

@Parameters:
Purpose: Defines parameters to pass to test methods.
Usage: Helps in parameterizing test methods by specifying parameters from testng.xml or data providers.

@DataProvider:
Purpose: Supplies data to test methods.
Usage: Provides test data to test methods, allowing parameterization of tests.

@Listeners:
Purpose: Listens to test events and performs actions.
Usage: Provides custom behavior based on test events like test failure, success, or skip.

How to skip a test? (Important Interview question)

@Test(enabled = false):
Purpose: Disables a specific test method.
Usage: used to skip a specific test method that's not ready or needs exclusion.

@Test(singleThreaded = true):
Purpose: Executes test methods in a single thread.
Usage: Ensures that the test method runs in isolation without parallel execution.

@Test(dependsOnMethods = { "methodName" }):
Purpose: Defines dependencies between test methods.
Usage: Ensures that a test method executes only after the specified method(s) complete successfully.

@Test(priority = n):
Purpose: Sets the priority order of test methods.
Usage: Defines the execution sequence by assigning priorities to test methods.

Note - Priorities can be negative, 0 and positive numbers. The smaller the number, higher is the priority. **(Famous Interview Question)**

@Listeners at the method level:
Purpose: Applies listeners to specific methods.
Usage: Attaches listeners to individual test methods for specialized handling.

Note - Often used in report generation for Test Execution Results.

@Test(invocationCount = n):
Purpose: Specifies the number of times a test method should be invoked.
Usage: Executes the same test method multiple times for load or stress testing.

@Test(threadPoolSize = n, invocationCount = m)
Purpose: Executes test methods concurrently with a specified thread pool size and invocation count.
Usage: Allows parallel execution of the same test method multiple times using a defined thread pool.

Code example -

```
@Test(threadPoolSize = 3, invocationCount = 10)
public void testMethod() {
   // Test logic executed concurrently with 3 threads and invoked 10 times
}
```

@DataProvider(parallel = true) for Parallel DataProviders

Purpose: Runs data providers in parallel.
Usage: Enables parallel execution of data providers.

Code example -

```
@DataProvider(parallel = true)
public Object[][] testData() {
    return new Object[][] { { "data1" }, { "data2" }, { "data3" } };
}
```

Interview questions in Next slides —-->

TestNG based Interview Questions

1) What is exception test?
   Ans: shared in previous slides

2) How to use data provider with parallel execution
   Ans: shared in previous slide

3) What is invocation count and threadpool size?
   Ans: shared in previous slides

4) What are tesNG test priority and what happens if priority is negative?
   Ans: shared in previous slide

5) What is AAA principle?

For more interview questions and in depth understanding use my courses, links available in my profile.

Also subscribe to my Medium and YouTube channels - Links in profile

Keep following - @Japneet Sachdeva - LinkedIN