In TestNG, **Listeners** are special classes that enable users to listen to various events and perform specific actions based on those events during test execution. Listeners can be used to monitor test execution, log results, generate custom reports, take screenshots on test failures, and handle custom actions dynamically. They act like event handlers, giving you a way to customize the behaviour of your tests based on their status.

Here's a breakdown of different types of Listeners in TestNG and how to implement them:

**1. Types of Listeners in TestNG**

TestNG provides several listeners, each with its unique functionalities:

* **ITestListener**: This listener listens to events related to test case execution, such as starting, passing, skipping, or failing of tests.
* **ISuiteListener**: Listens to events before and after the execution of a TestNG suite.
* **IInvokedMethodListener**: Monitors method invocations, allowing you to perform actions before and after any method (test, configuration) execution.
* **IReporter**: Used for generating custom reports after the test suite completes.
* **IConfigurationListener**: Monitors configuration methods' events (like @BeforeSuite, @AfterClass, etc.).
* **IAnnotationTransformer**: Allows modifications of TestNG annotations at runtime.
* **IHookable**: Controls the invocation of test methods.

**2. Implementing Listeners at Class Level**

To apply a listener to a specific class, you use the @Listeners annotation at the class level.

import org.testng.annotations.Listeners;

@Listeners(MyListenerClass.class)

public class MyTest {

// test methods

}

This way, the specified listener (MyListenerClass) will be applied only to this test class.

**3. Implementing Listeners at Suite Level**

To apply listeners at the suite level, you can configure them directly in the **TestNG XML file**:

xml

<suite name="SuiteName">

<listeners>

<listener class-name="com.example.MyListenerClass" />

</listeners>

<test name="TestName">

<classes>

<class name="com.example.MyTest" />

</classes>

</test>

</suite>

Listeners defined in the suite apply to all tests and classes within that suite.

**4. ITestListener Methods: onStart() and onFinish()**

Two of the most commonly used methods in ITestListener are:

* **onStart()**: Executes before any test starts within the suite.
* **onFinish()**: Executes after all tests have finished within the suite.

Example usage:

import org.testng.ITestContext;

import org.testng.ITestListener;

public class MyListener implements ITestListener {

@Override

public void onStart(ITestContext context) {

System.out.println("Starting test: " + context.getName());

}

@Override

public void onFinish(ITestContext context) {

System.out.println("Finished test: " + context.getName());

}

// Other methods like onTestSuccess, onTestFailure, etc.

}

**5. Difference Between Listeners and Annotations**

* **Listeners**: Allow you to monitor or modify test behavior based on specific events (like test failure, start, or finish).
* **Annotations**: Define specific points where test actions take place (e.g., @BeforeMethod, @AfterSuite). They control the flow of test setup and teardown.

**6. TestListenerAdapter Class**

The TestListenerAdapter is a simple implementation of the ITestListener interface, allowing users to override only the necessary methods without implementing every single method in ITestListener.

Example:

import org.testng.TestListenerAdapter;

public class MyTestListener extends TestListenerAdapter {

@Override

public void onTestFailure(ITestResult result) {

System.out.println("Test failed: " + result.getName());

// Screenshot logic can go here

}

}

**7. Taking Screenshots for Failed Tests Using Listeners**

You can take screenshots on failed tests by implementing ITestListener and overriding the onTestFailure() method.

@Override

public void onTestFailure(ITestResult result) {

// Code to capture screenshot

System.out.println("Taking screenshot for failed test: " + result.getName());

}

**8. Retrieving Test Tag Name along with Test Method Name**

To retrieve the test tag name along with the test method name, use the ITestContext and ITestResult in methods like onTestStart or onTestFailure.

@Override

public void onTestStart(ITestResult result) {

String testName = result.getMethod().getMethodName();

String suiteName = result.getTestContext().getSuite().getName();

System.out.println("Running test: " + suiteName + " - " + testName);

}

**9. Taking Screenshots for Failed Tests Using Dependency Injection**

Using dependency injection, you can pass WebDriver objects to your listener methods and take screenshots. First, define the listener and use @Listeners to enable it in your test class.

In onTestFailure():

@Override

public void onTestFailure(ITestResult result) {

Object currentClass = result.getInstance();

WebDriver driver = ((YourTestClass) currentClass).getDriver();

// Use the driver to take a screenshot here

}

This setup ensures you can use the same WebDriver instance for taking screenshots.