TestNG Listeners

Sure! Let's dive deep into **TestNG Listeners** – one of the most powerful features of **TestNG** for customizing test execution flow.

✅ What are TestNG Listeners?

Listeners in TestNG **listen** to events occurring during the test execution (like test start, test pass/fail, suite start/finish, etc.) and allow us to **perform actions automatically** in response to those events.

🎯 Why Use Listeners?

Listeners help to:

- Take screenshots on failure
- Log test status to reports
- Send emails after test execution
- Implement retry mechanisms
- Track test progress

🔰 Types of Listeners in TestNG

Listener Interface	Purpose
`ITestListener`	Tracks test method events like start, success, failure, skip
`ISuiteListener`	Tracks events before/after the test suite
`IClassListener`	Listens to class-level test events
`IInvokedMethodListener`	Captures before/after any method (test/configuration)
`IReporter`	Generates custom test reports
`IRetryAnalyzer`	Retries failed tests

🚀 Most Commonly Used: `ITestListener`

📦 Methods in `ITestListener`

```
java

public interface ITestListener {
    void onTestStart(ITestResult result);
    void onTestSuccess(ITestResult result);
    void onTestFailure(ITestResult result);
    void onTestSkipped(ITestResult result);
    void onTestFailedButWithinSuccessPercentage(ITestResult result);
    void onStart(ITestContext context);
```

```
    void onFinish(ITestContext context);
}
```

📘 Basic Example: Logging Events with Listener

1️⃣ Create a Listener Class

```
java

import org.testng.ITestContext;
import org.testng.ITestListener;
import org.testng.ITestResult;

public class MyTestListener implements ITestListener {

    @Override
    public void onTestStart(ITestResult result) {
        System.out.println("Test Started: " + result.getName());
    }

    @Override
    public void onTestSuccess(ITestResult result) {
        System.out.println("Test Passed: " + result.getName());
    }

    @Override
    public void onTestFailure(ITestResult result) {
        System.out.println("Test Failed: " + result.getName());
    }

    @Override
    public void onTestSkipped(ITestResult result) {
        System.out.println("Test Skipped: " + result.getName());
    }

    @Override
    public void onStart(ITestContext context) {
        System.out.println("Test Suite Started: " + context.getName());
    }

    @Override
    public void onFinish(ITestContext context) {
        System.out.println("Test Suite Finished: " + context.getName());
    }
}
```

2️⃣ Register the Listener

You can register the listener in **3 ways**:

✅ Method 1: Using `@Listeners` Annotation

```
java

import org.testng.annotations.Listeners;
import org.testng.annotations.Test;

@Listeners(MyTestListener.class)
public class MyTest {

    @Test
    public void testPass() {
```

```
        System.out.println("Inside testPass");
    }

    @Test
    public void testFail() {
        System.out.println("Inside testFail");
        assert false;
    }
}
```

✅ Method 2: In `testng.xml`

```
xml


<listeners>
    <listener class-name="MyTestListener"/>
</listeners>
```

✅ Method 3: Programmatically

```
java


TestNG testng = new TestNG();
testng.setTestClasses(new Class[]{MyTest.class});
testng.addListener(new MyTestListener());
testng.run();
```

📸 Example: Take Screenshot on Test Failure

Let's say you have a utility to take a screenshot.

```
java


public class ScreenshotUtil {
    public static void capture(String testName) {
        System.out.println("Screenshot captured for test: " + testName);
        // Here you'd use WebDriver to capture screenshots
    }
}
```

Use this in the listener:

```
java


@Override
public void onTestFailure(ITestResult result) {
    ScreenshotUtil.capture(result.getName());
}
```

🔁 Retry Failed Tests: `IRetryAnalyzer`

```
java


import org.testng.IRetryAnalyzer;
import org.testng.ITestResult;
```

```
public class RetryAnalyzer implements IRetryAnalyzer {
    private int count = 0;
    private static final int maxTry = 2;

    public boolean retry(ITestResult result) {
        if (count < maxTry) {
            count++;
            return true;
        }
        return false;
    }
}
```

In your test:

```
java

@Test(retryAnalyzer = RetryAnalyzer.class)
public void testRetry() {
    System.out.println("Running testRetry");
    assert false;
}
```

📊 Custom Report: `IReporter`

```
java

import org.testng.IReporter;
import org.testng.ISuite;
import org.testng.xml.XmlSuite;
import java.util.List;

public class MyReporter implements IReporter {
    public void generateReport(List<XmlSuite> xmlSuites, List<ISuite> suites, String
outputDirectory) {
        System.out.println("Custom Report Generated in: " + outputDirectory);
    }
}
```

Add to **`testng.xml`**:

```
xml

<listeners>
    <listener class-name="MyReporter"/>
</listeners>
```

🔪 Example Output

```
yaml

Test Suite Started: MyTest
Test Started: testPass
Inside testPass
Test Passed: testPass
Test Started: testFail
Inside testFail
Test Failed: testFail
```

```
Test Suite Finished: MyTest
Screenshot captured for test: testFail
```

✅ Summary

Interface	Purpose
`ITestListener`	Track test start, success, fail, skip
`ISuiteListener`	Actions at suite start/end
`IRetryAnalyzer`	Retry failed test logic
`IReporter`	Custom reporting

💡 Assignment for You

1. Create a listener that logs the test name and execution time for each test.
2. Enhance the listener to retry failed tests.
3. Capture a dummy screenshot only on failure.
4. Create a custom reporter that lists how many tests passed, failed, and skipped.

Would you like a **mini-project combining all of this** into a reusable TestNG framework with listeners, retry logic, and screenshot logic?