A **TestNG Retry Listener** is a feature in TestNG that allows you to automatically retry failed tests a specified number of times before marking them as failed. This is useful in cases where tests might occasionally fail due to environmental issues, network fluctuations, or other intermittent issues, rather than a defect in the code itself.

In TestNG, you can implement a retry mechanism by creating a custom retry analyzer using the IRetryAnalyzer interface. This interface provides the flexibility to define a retry policy for failed tests. Let's go through the setup and usage of a TestNG retry listener.

**1. Implementing a Retry Analyzer**

To create a retry listener, you need to implement the IRetryAnalyzer interface, which has a single method retry(ITestResult result):

import org.testng.IRetryAnalyzer;

import org.testng.ITestResult;

public class RetryAnalyzer implements IRetryAnalyzer {

private int retryCount = 0;

private static final int maxRetryCount = 3;

@Override

public boolean retry(ITestResult result) {

if (retryCount < maxRetryCount) {

retryCount++;

return true; // Test will be retried

}

return false; // Test will not be retried

}

}

In this example:

* retryCount tracks the number of attempts.
* maxRetryCount specifies the maximum number of retries allowed.

**2. Applying the Retry Analyzer to a Test Method**

To apply the retry analyzer to specific test methods, use the @Test annotation's retryAnalyzer attribute:

import org.testng.annotations.Test;

public class MyTest {

@Test(retryAnalyzer = RetryAnalyzer.class)

public void testMethod() {

// Test code here

}

}

This setup will automatically retry testMethod up to 3 times if it fails.

**3. Applying Retry Analyzer Globally (Using Listeners)**

If you want to apply the retry mechanism to all test methods in a suite or across classes, you can use an **annotation transformer**. This will allow you to set the retry analyzer globally instead of specifying it on each test method.

import org.testng.IAnnotationTransformer;

import org.testng.IRetryAnalyzer;

import org.testng.annotations.ITestAnnotation;

public class RetryListener implements IAnnotationTransformer {

@Override

public void transform(ITestAnnotation annotation, Class testClass,

Constructor testConstructor, Method testMethod) {

IRetryAnalyzer retry = annotation.getRetryAnalyzer();

if (retry == null) {

annotation.setRetryAnalyzer(RetryAnalyzer.class);

}

}

}

Then, add this listener in your testng.xml file to apply it to the entire suite:

xml

<listeners>

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

</listeners>

Or you can add it using the @Listeners annotation at the class level:

import org.testng.annotations.Listeners;

@Listeners(RetryListener.class)

public class MyTest {

// Test methods

}

**4. Benefits of a Retry Listener**

* **Stabilizes Test Suites**: Retries prevent flaky tests from causing suite failures due to transient issues.
* **Less Manual Intervention**: Reduces the need to rerun failed tests manually, which can save time in CI/CD pipelines.
* **Customizable Retry Logic**: You can set specific retry counts or even conditions to retry based on specific exceptions or test outcomes.

**5. Example Use Case**

Suppose a test depends on external resources (like APIs or databases) that occasionally face network issues. A retry mechanism can catch these intermittent failures by re-running the test, improving the reliability of test results in these cases.