**TestNG Listeners**

There are many types of listeners available in TestNG for example IAnnotationTransformer, IAnnotationTransformer2, IConfigurable, IConfigurationListener, IConfigurationListener2, IExecutionListener, IHookable, IInvokedMethodListener, IInvokedMethodListener2, IMethodInterceptor, IReporter, ISuiteListener, ITestListener.

As far as testing concern only few can be used effectively such as :

**ISuiteListener**: It has two method in it **onStart()** & **onFinish()**. Whenever a class implements this listener, TestNG guarantees the end-user that it will invoke the methods onStart() and onFinish() before and after running a TestNG Suite. So before TestNG picks up your suite for execution, it first makes a call to onStart() method and runs whatever has been scripted in this method. In a similar way, it again makes a call to onFinish() method after a suite has been run.

**ITestListener**: The working of this listener is also exactly the same as ISuiteListerner but the only difference is that it makes the call before and after the Test not the Suite. It has seven methods in it.

**onFinish()**: Invoked after all the tests have run and all their Configuration methods have been called.

**onStart():**Invoked after the test class is instantiated and before any configuration method is called.

**onTestFailedButWithinSuccessPercentage(ITestResult result):**Invoked each time a method fails but has been annotated with successPercentage and this failure still keeps it within the success percentage requested.

**onTestFailure(ITestResult result):** Invoked each time a test fails.

**onTestSkipped(ITestResult result):**Invoked each time a test is skipped

**onTestStart(ITestResult result):**Invoked each time before a test will be invoked.

**onTestSuccess(ITestResult result):**Invoked each time a test succeeds.

**IInvokedMethodListener**: The working of this listener is also exactly the same as ISuiteListerner & ITestListerner and the only difference is that it makes the call before and after every Method. It has only two methods in it.

**afterInvocattion():** Invoke after each method

**beforeInvocation():** Invoke before each method

LoggingClass

import org.testng.annotations.Test;

public class LoggingClass {

@Test(priority = 0)

public void methodAddingNumbers() {

System.out.println("Helloo.. Im in method adding numbers");

}

@Test(priority = 1)

public void dividedByZero() {

System.out.println("Helloo.. Im in method divided by zero");

int e = 1 / 0;

}

@Test(dependsOnMethods = { "dividedByZero" })

public void methodSkip() {

System.out.println("Helloo.. Im in method skip");

}

}

Listener

import org.testng.IClass;

import org.testng.ITestResult;

import org.testng.TestListenerAdapter;

public class ListenerClass extends TestListenerAdapter {

@Override

public void onTestStart(ITestResult tr) {

log("Test Started....");

}

@Override

public void onTestSuccess(ITestResult tr) {

log("Test '" + tr.getName() + "' PASSED");

// This will print the class name in which the method is present

log(tr.getTestClass());

// This will print the priority of the method.

// If the priority is not defined it will print the default priority as

// 'o'

log("Priority of this method is " + tr.getMethod().getPriority());

System.out.println(".....");

}

@Override

public void onTestFailure(ITestResult tr) {

log("Test '" + tr.getName() + "' FAILED");

log("Priority of this method is " + tr.getMethod().getPriority());

System.out.println(".....");

}

@Override

public void onTestSkipped(ITestResult tr) {

log("Test '" + tr.getName() + "' SKIPPED");

System.out.println(".....");

}

private void log(String methodName) {

System.out.println(methodName);

}

private void log(IClass testClass) {

System.out.println(testClass);

}

}

There are essentially two ways of adding up a listener to a particular class.

1) Implement TestNG Listener to your test class

|  |  |
| --- | --- |
|  | package automationFramework;    import org.testng.annotations.AfterMethod;    import org.testng.annotations.BeforeMethod;    import org.testng.annotations.Test;    // This code will implement TestNG listeners    @Listeners(PackageName.ListenerClassName)    // For e.g. @Listeners(utility.Listener.class)    public class TestListener {      @Test      public void main() {      }    } |

2) Listener tag in TestNG xml: Although approach 1 is more than enough to get you started, it’s not an “elegant” way of using Listeners, because you are forced to add this @Listeners section to each of your classes, which you perhaps won’t want. So what you do is, you create a TestNG Suite xml and then add up the listeners section to this suite xml file. That way, all of your tests would essentially leverage the listener that you wrote.

XML – Configuration

<!DOCTYPE suite SYSTEM "[http://testng.org/testng-1.0.dtd"](http://testng.org/testng-1.0.dtd) >

<suite name="Log Suite Example" verbose="1">

<listeners>

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

</listeners>

<test name="TestNG logs sample" preserve-order="true">

<classes>

<class name="com.example.logging.LoggingClass">

<methods>

<include name="methodAddingNumbers" />

<include name="dividedByZero" />

<include name="methodSkip" />

</methods>

</class>

</classes>

</test>

</suite>