TestNG Listeners

It is quite an important and the last chapter of my tutorial on TestNG. It took me a long time to figure out the perfect simple example for my viewers, as I am from a non-technical background and I hooped around so many sites to get the perfect example of listener but I was unable to

find one. Everybody has pasted almost same example all over the internet. Enough talking, let's start now.

In very non-technical term, TestNG manages everything through Suite, Test and Methods and the Listeners gives us the ability to act before and after of every Suite, Test and Methods.

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

Ok, so now am guessing that you must be pretty aware of what listeners are all about and we also saw how to write a listener. Now comes the big question.

How to do it…

1) Create a '***New Class***' file and give it a name '**Listener**', by right click on the Package and select **New** > **Class**.

2) Now Implements ISuiteListener, ITestListener and IInvokedMethodListener to this newly created class. For implementing a listener class, the class has to implement the *org.testng.ITestListener interface. These classes are notified at runtime by TestNG when the test starts, finishes, fails, skips, or passes.*

```
package utility;

import org.testng.IInvokedMethod;

import org.testng.IInvokedMethodListener;

import org.testng.ISuite;

import org.testng.ISuiteListener;

import org.testng.ITestContext;

import org.testng.ITestListener;

import org.testng.ITestNGMethod;

import org.testng.ITestResult;

import org.testng.Reporter;
```

```
20
21 public class Listener implements ITestListener, ISuiteListener, IInvokedMethodListener {
22
23  // This belongs to ISuiteListener and will execute before the Suite start
24
25  @Override
26
27  public void onStart(ISuite arg0) {
28
29  Reporter.log("About to begin executing Suite " + arg0.getName(), true);
30
31  }
32
33  // This belongs to ISuiteListener and will execute, once the Suite is finished
34
35  @Override
36
37  public void onFinish(ISuite arg0) {
38
39  Reporter.log("About to end executing Suite " + arg0.getName(), true);
40
41  }
42
43  // This belongs to ITestListener and will execute before starting of Test set/batch
44
45  public void onStart(ITestContext arg0) {
46
47  Reporter.log("About to begin executing Test " + arg0.getName(), true);
48
49  }
50
51  // This belongs to ITestListener and will execute, once the Test set/batch is finished
52
53  public void onFinish(ITestContext arg0) {
54
55  Reporter.log("Completed executing test " + arg0.getName(), true);
56
57  }
58
59  // This belongs to ITestListener and will execute only when the test is pass
60
61  public void onTestSuccess(ITestResult arg0) {
62
63  // This is calling the printTestResults method
64
65  printTestResults(arg0);
66
67  }
68
69  // This belongs to ITestListener and will execute only on the event of fail test
70
71  public void onTestFailure(ITestResult arg0) {
72
73  // This is calling the printTestResults method
74
75  printTestResults(arg0);
76
77  }
78
79  // This belongs to ITestListener and will execute before the main test start (@Test)
80
81  public void onTestStart(ITestResult arg0) {
82
83  System.out.println("The execution of the main test starts now");
84
85  }
86
87  // This belongs to ITestListener and will execute only if any of the main test(@Test) get skipped
88
89  public void onTestSkipped(ITestResult arg0) {
90
91  printTestResults(arg0);
92
93  }
94
95  public void onTestFailedButWithinSuccessPercentage(ITestResult arg0) {
96
97  }
98
99  // This is the method which will be executed in case of test pass or fail
100
101  // This will provide the information on the test
102
103  private void printTestResults(ITestResult result) {
104
105  Reporter.log("Test Method resides in " + result.getTestClass().getName(), true);
106
107  if (result.getParameters().length != 0) {
108
109  String params = null;
```

```

 for (Object parameter : result.getParameters()) {

 params += parameter.toString() + ",";

 }

 Reporter.log("Test Method had the following parameters : " + params, true);

 }

 String status = null;

 switch (result.getStatus()) {

 case ITestResult.SUCCESS:

 status = "Pass";

 break;

 case ITestResult.FAILURE:

 status = "Failed";

 break;

 case ITestResult.SKIP:

 status = "Skipped";

 }

 Reporter.log("Test Status: " + status, true);

 }

 // This belongs to IInvokedMethodListener and will execute before every method including @Before @After @Test

 public void beforeInvocation(IInvokedMethod arg0, ITestResult arg1) {

 String textMsg = "About to begin executing following method : " + returnMethodName(arg0.getTestMethod());

 Reporter.log(textMsg, true);

 }

 // This belongs to IInvokedMethodListener and will execute after every method including @Before @After @Test

 public void afterInvocation(IInvokedMethod arg0, ITestResult arg1) {

 String textMsg = "Completed executing following method : " + returnMethodName(arg0.getTestMethod());

 Reporter.log(textMsg, true);

 }

 // This will return method names to the calling function

 private String returnMethodName(ITestNGMethod method) {

 return method.getRealClass().getSimpleName() + "." + method.getMethodName();

 }

}
```

3) Create a '***New Class***' file and give it a name '**TestListener**', by right click on the Package and select **New** > **Class**.

```
package automationFramework;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

public class TestListener {

  @Test

  public void main() {

   System.out.println("Execution of Main test is carring on");

  }

  @BeforeMethod

  public void beforeMethod() {
```

```

    System.out.println("Execution of Before method is carring on");

  }

  @AfterMethod

  public void afterMethod() {

    System.out.println("Execution of After method is carring on");

  }

}
```

How do I let TestNG know that I have such a listener which it should invoke when it is executing my tests ?

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.

```
<suite name="Suite-Listeners" parallel="none">

 <listeners>

 <listener class-name="utility.Listener"></listener>

 </listeners>

 <test name="Batch-Listeners">

 <classes>

 <class name="automationFramework.TestListener" />

 </classes>

 </test>

</suite>
```

Output of the test case will look like this:

For more updates on TestNG Tutorial, please Subscribe to our Newsletter.

Please ask any questions on ForumsQA, in case of any issues or doubts.

Category: Selenium-Webdriver • By Lakshay Sharma • April 27, 2014

Tags: TestNG

Share this post

Author: Lakshay Sharma

http://toolsqa.wpengine.com

I'M LAKSHAY SHARMA AND I'M A TEST AUTOMATION ENGINEER.Have passed 11 years playing with automation in mammoth projects like O2 (UK), Sprint (US), TD Bank (CA), Canadian Tire (CA), NHS (UK) & ASOS(UK). Currently I am working with **BLOOMREACH** as SDET.I am passionate about designing Automation Frameworks that are effective and easy to maintain. For automating websites my weapons are QTP and Selenium (WebDriver JAVA & C#).I live in Amsterdam(NL), with my wife and a lovely daughter. Please connect with me at **LinkedIn** or follow me on **Instagram**.

Related posts

Common Exceptions in Selenium WebDriver
October 15, 2018

Inspect Elements with Chrome Developer Tools
October 1, 2018

WebDriverManager
September 29, 2018

Download File using Selenium and Verifying
November 27, 2017

@CacheLookup in PageObjectModel
November 12, 2017

How to Verify file downloaded successfully through selenium using IIS Service
June 10, 2017

SUBSCRIBE TO NEWSLETTER

Enter your email address:

Subscribe

GOT SELENIUM PROBLEMS ?

RECENT POST

Different types of Asserts in Postman

10 Simple Tips to Approach API Testing for Beginners

How to Handle Scroll to Element in Mobile Automation with Katalon Studio

Cookies in Postman

What is a Cookie?

Sessions In Postman

OAuth 2.0 Authorization with Postman

OAuth 2.0 Authorization

A Guide on Integrating Katalon Studio with TestRail

Using Katalon Studio to Approach Web Element Locators

Mock Server in Postman

Automate Shadow DOM Elements with Katalon Studio

Postman Cheat Sheet

Postman Interview Questions

REST API & WebServices Testing with Katalon Studio

Got a Question?

.

.Subscribe in a reader

.

Site Links

Selenium Training
Corporate Training
Video Tutorials
About Us
Guest Blogs
Testimonials
Contact Us
SITEMAP

Tutorials

- **Software Testing**
- **Selenium - Java**
- **Selenium - C#**
- **Cucumber**
- **SpecFlow**
- **Appium**
- **TestNg**
- **JUnit**
- **Maven**
- **Java**
- **Postman**
- **Katalon**

Author

I'M LAKSHAY SHARMA AND I'M A TEST AUTOMATION ENGINEER.

Have passed 11 years playing with automation in mammoth projects like O2 (UK), Sprint (US), TD Bank (CA), Canadian Tire (CA), NHS (UK) & ASOS(UK).
Currently I am working with **BLOOMREACH** as SDET.

I am passionate about designing Automation Frameworks that are effective and easy to maintain. For automating websites my weapons are **QTP** and **Selenium (Webdriver)**. I live in Amsterdam(NL), with my wife and a lovely daughter.

Please connect with me at **LinkedIn** or follow me on **Instagram**.