## What are TestNG Listeners?

The first thing that comes to mind by reading the term "***listeners***" is that it must be listening to something in the code and being a "***good listener***", it does. ***TestNG listeners are the piece of code that listens to the events occurring in the TestNG***. If the event matches the event for which we want the listener to listen, it executes the code, which ultimately results in modifying the default behavior of TestNG. For example, we want to print the exception error onto the reports only if the test fails. Here, we can apply a TestNG listener that will listen to the event of "***failing of test case***" and when it does, it will log the error.

To save the unnecessary utilization of the resources, we cannot let the listeners listen to the code all the time. Therefore, the listeners in TestNG are activated either before the test execution or before (noticing that test execution takes the majority share of testing time). ***TestNG Listeners are applied as interfaces in the code*** because "***Listeners***" is a "***class***" in TestNG. TestNG provides us with loads of listeners. We will discuss these in the next section.

**Types Of Listeners In TestNG**

TestNG provides a bunch of listeners as a part of its testing environment. These listeners are as follows:

1. *IAnnotationTransformer*
2. *IExecutionListener*
3. *IHookable*
4. *IInvokedMethodListener*
5. *IMethodInterceptor*
6. *IReporter*
7. *ISuiteListener*
8. *ITestListener - Done*
9. *IConfigurable*
10. *IConfigurationListener*
11. *IInvokedMethod*
12. *IConfigurableListener*
13. *IExecution*
14. *IAnnotationTransformer2*
15. *IInvokedMethodIntercepter2*
16. *IInvokedMethodIntercepter*

**ITestListener In TestNG**

*ITestListener* is the most used listener in TestNG with [***Selenium webdriver***](https://www.toolsqa.com/selenium-webdriver/selenium-tutorial/). The *ITestListener* ***implements*** since it is an interface, and the class in which we implement the listener overrides the *ITestListener* methods. *ITestListener* listens to specific events (*depending on its methods*) and executes the code written inside the method.  With *ITestListener* in TestNG, we can also log the events onto the reports using the Selenium web driver. The *ITestListener* contains the following methods:

* ***onStart***: *This method invokes when the test class is instantiated and before executing any test method*.

***Syntax***: ***void onStart(ITestContext context);***

* ***onFinish***: *This method invokes when all the test methods have run, and calling of all of their configuration methods happens*.

***Syntax***: ***void onFinish(ITestContext context);***

* ***onTestStart***:  *This method invokes every time a test method is called and executed*.

***Syntax***: ***void onTestStart(ITestResult result);***

* ***onTestSuccess***: *This method is invoked every time a test case passes (succeeds)*.

***Syntax***: ***void onTestSuccess(ITestResult result);***

* ***onTestFailure***: *This method invokes every time a test case fails*.

***Syntax***: ***void onTestFailure(ITestResult result);***

* ***onTestSkipped***: *This method invokes every time a test skips*.

***Syntax***: ***void onTestSkipped (ITestResult result);***

* ***onTestFailedButWithinSuccessPercentage***: *This method invokes when the test method fails as a whole but has passed a certain success percentage, which is defined by the user*.

***Syntax***: ***void onTestFailedButSuccessPercentage (ITestResult result);***

Now in the above syntaxes, you must be wondering about the words ***ITestContext*** and ***ITestResult***. So, the term ***'ITestResult' is an interface that describes the result of the test***. Therefore '*result*' has been passed as its instance in the syntax. Whereas ' *ITestContext* ' ***is a class that defines an instance*** '*context*', ***which contains all the information about the test run***. We can use this information to pass to our listeners, and they can proceed with their queries.

***How to implement ITestListener in TestsNG?***

This section will explain step by step how to use listeners in TestNG to invoke various functions. It is important to note here that ***Listeners can implement in two ways in TestNG***:

* ***At the class level***: *Annotating listeners on each class in the test code*.
* ***At the suite level***: *Define the class names to implement listeners in the TestNG XML file*.

**IAnnotationListener**

IAnnotationTransformer transforms the [TestNG annotations](https://www.browserstack.com/guide/testng-annotations-in-selenium) at run time. A scenario may appear in which the user seeks to override the content of the annotation based on a condition. In such a case it is not necessary to make changes in the source code.

**IExecutionListener**

As the name suggests, it monitors the beginning and end of TestNG execution. This listener is mostly used to start/stop the server while starting or ending code execution. It may also be used to inform respective stakeholders via email that execution shall start or when it ends. It has two methods:

1. ***onExecutionStart()*** – invoked before TestNG starts executing the suites
2. ***onExecutionFinish()*** – invoked after all TestNG suites have finished execution

IHookable

If a class implements this interface, its run method will be invoked instead of each test method.

Using the callback method of the IHookCallBack parameter, the invocation of the test method can be performed.

It has a single method name run, which accepts two parameters.run(IHookCallBack callBack, ITestResult testResult)

Simply it means that, the value which is mentioned in listener will not run and other values run other than that.

IInvokedMethodListener

This listener gets invoked before and after a method in TestNG. These methods constitute both test and other configuration methods. These listeners are useful for setting up configuration or other cleanup activities. It contains two methods:

1. **beforeInvocation()**: this method gets invoked before every method
2. **afterInvocation()**: this method gets invoked after every method