iamneo

# TestNG Data Provider and Parallel Testing

# TestNG Listeners

- *TestNG listeners are the piece of code that listens to the events occurring in the 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.
- *TestNG Listeners are applied as interfaces in the code* because "*Listeners*" is a "*class*" in TestNG.

# Types of TestNG Listeners

- TestNG provides a bunch of listeners as a part of its testing environment. These  important listeners are as follows
    - *ITestListener* :*TestListener* listens to specific events (*depending on its methods*) and executes the code written inside the method.
    - IReporter Listeners :The *IReporter* interface in the TestNG Listener provides us with a medium to generate custom reports (*i.e., customize the reports generated by TestNG*).
    - ISuiteListener :it listens to the event of the start of a suite execution and end of the suite execution. *ISuiteListener* then runs the methods only before the start of the suite and at the end.

# ITestListener TestNG Listeners

- **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);**

# ITestListener TestNG Listeners

- **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);**

# ISuite TestNG Listeners

- **onStart:** This method invokes before the test suite execution starts.

  Syntax: **void onStart(ISuite suite);**

- **onFinish:** This method invokes after the test suite execution ends.

  Syntax: **void onFinish(ISuite suite);**

# IReporter TestNG Listeners

- **getPassedTests():** the number of tests that have passed

  for eg: tc.getPassedTests().getAllResults().size()

- **getFailedTests():** the number of tests that have failed

  for eg: tc.getFailedTests().getAllResults().size()

- **getSkippedTests():** the number of skipped tests.

  for eg: tc.getSkippedTests().getAllResults().size()

# TestNG Data provider

- When you need to pass complex parameters or parameters that need to be created from Java (*complex objects, objects read from a property file or a database, etc...*), in such cases parameters can be passed using *Dataproviders*.

- A Data Provider returns an array of objects.

# Steps to Create a TestNG Data provider

- Create a TestNG class '*DataProviderTest*' by Pressing Ctrl+N, select '*Create TestNG Class*' under TestNG category and Under Annotations, check '*DataProvider*' and click Finish.
- By default, the DataProvider name is '*dp*', change it to as per your requirement. This method returns array of object array.
- Add a method to your Test class. This method takes two strings as input parameters.
- Write script for LogIn Application under method `@Test`.

# Create a Test Datasheet

1. Create a '*New Package*' file and name it as '*testData*', by right click on the Project and select New > Package. I always place my Test Data file under a separate test data folder.
2. Place an *Excel* file in the above-created package location and save it as "*TestData.xlsx*".

# Create functions to Open & Read data from Excel

We need a way to open this Excel sheet and read data from it within our Selenium test script.

For this purpose, I use the Apache POI library, which allows you to read, create and edit Microsoft Office-documents using Java.

The classes and methods we are going to use to read data from Excel sheet are located .

# Accepting data from Excel using Data Provider

1. Create a TestNG class '*DataProviderWithExcel*' by Pressing Ctrl+N , select '*Create TestNG Class*' under TestNG category and Under Annotations, check *@BeforeMethod*, *@AfterMethod* & *DataProvider*and click Finish.
2. Add a method to your Test class. This method takes two strings as input parameters.
3. Now divide the test case into three parts :

    @BeforeMethod

    @Test

    @AfterMethod

# Parallel Testing

- *Parallel testing or parallel execution, as the name suggests, is a process of running the test case parallelly rather than one after the other*.
- *Parallel execution would give us the correct idea of the stability and performance of the software much faster than running serially*.
- Parallel testing is used heavily with *Selenium* because of the importance of cross-browser testing in the market today.

# *Advantages of Parallel Testing*

- *Reduces Time: Running the tests in parallel reduces the overall execution time.*
- *Allow Multi-Threaded Tests: Using the parallel execution in TestNG, we can allow multiple threads to run simultaneously on the test case providing independence in the execution of different components of the software.*

# *Where to apply Parallel Testing*

Parallel testing must be used with the test case methods to run them in parallel TestNG offers four more areas where we can go ahead with parallel testing.

- *Methods*: *This will run the parallel tests on all* `@Test` *methods in TestNG*.
- *Tests*: *All the test cases present inside the <test> tag will run with this value*.
- *Classes*: *All the test cases present inside the classes that exist in the XML will run in parallel*.
- *Instances*: *This value will run all the test cases parallelly inside the same instance*.

# *Running test classes Parallelly in TestNG using Selenium*

***ChromeTest.java.***

```
@BeforeTest
        public void beforeTest() {
                System.out.println("Initilizing the Google Chrome Driver");
                driver = new ChromeDriver();  }
       @Test
        public void ChromeTestMethod()
        {
          //Initialize the chrome driver
         System.out.println("The thread ID for Chrome is "+ Thread.currentThread().getId());
          driver.get("https://www.facebook.com");
          driver.findElement(By.name("login")).click(); }
      @AfterTest
      public void afterTest() {
         System.out.println("Closing the browser ");
         driver.close(); }
```

# *Running test classes Parallelly in TestNG using Selenium*

**FirefoxTest.java**

```
@BeforeTest
        public void beforeTest() {
                System.out.println("Initilizing the Google Chrome Driver");
                driver = new GeckoDriver();  }
       @Test
        public void FirefoxTestMethod()
        {
          //Initialize the chrome driver
         System.out.println("The thread ID for Chrome is "+ Thread.currentThread().getId());
          driver.get("https://www.facebook.com");
          driver.findElement(By.name("login")).click(); }
      @AfterTest
      public void afterTest() {
         System.out.println("Closing the browser ");
         driver.close(); }
```

# *Parallel Testing using XML*

We just need to parallelize them in the XML file.

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >
<suite name = "Parallel Testing Suite">
  <test name = "Parallel Tests" parallel = "classes" thread-count = "2">
    <classes>
      <class name = "ChromeTest" />
      <class name = "FirefoxTest" />
    </classes>
  </test>
</suite>
```