TEST AUTOMATION UNIVERSITY
Rex Jones II

Chapter 6.1: Introduction To TestNG Assertions

Introduction

I have a question for you. Up to this point, what have we tested. We have automated Opening Chrome and the application, signing into the application, searching for a user, signing out, closing chrome, and closing the application. We have not verified our Test Script and not sure if it truly Passed or Failed. The Console shows Passed and the Results tab shows all 3 Test Passed. However, it shows Passed because there was no error in our automation code and no error when running our Automation Test Script. Let me show you something else. I'm going to remove all code to search for a user then Run. The Test Script did not search for a user but look what the Console shows PASSED: userSearch although Number 3 Search For User is missing. The Results tab also shows Passed for userSearch. That's not right.

What Are TestNG Assertions

What Are TestNG Assertions? TestNG Assertions verify if our test truly Passed or Failed. It's a line of code that is placed in our Test Method to verify a condition.

TestNG Assertion Methods

TestNG Assertion Methods. There are many assertions for TestNG but most of them are overloaded versions of the following methods: assertTrue, assertFalse, assertSame, assertNotSame, assertNotNull, and assertEquals. Generally, all of these TestNG Assertions have the same 3 parameters: Actual Result, Expected Result, and a String. It's the same with JUnit Assertions which have an assertion class located in TestNG's distribution.

JUnit & TestNG Assertions

JUnit has a class called junit.framework.Assert that have similar overloaded methods. TestNG turned around and added the same JUnit class called org.testng.AssertJUnit to its distribution. Why did TestNG add the same class as JUnit? TestNG added the same class as JUnit to guarantee all assertions keep working if we migrate our test from JUnit to TestNG. TestNG also added another class called org.testng.Assert.

The main difference between JUnit's class and TestNG's class is the syntax. Their parameters are available in reverse order. For example, the assertEquals method for JUnit has a String as the first parameter followed by an expected result then an actual result. The same method for TestNG has actual as the first parameter, expected as the second parameter then String as the last parameter.

Let's go to Eclipse and I'll show you the assertions. Go to the TestNG Library, maximize the TestNG jar file, select org.testng package, and we see both classes: Assert and AssertJUnit class. Maximize Assert and there's a lot of methods. Maximize AssertJUnit and we some of the same overloaded methods. The methods within our Assert class are considered Hard Asserts. In the next chapter, we will cover Hard Asserts.