TEST AUTOMATION UNIVERSITY

Rex Jones II

Chapter 5:
TestNG Assertions

Table of Contents

Chapter 5.1 – Introduction To TestNG Assertions

Introduction

Hello and Welcome To Chapter 5, TestNG Assertions. This chapter is a series of 3 videos. The videos are Chapter 5.1 - Introduction To TestNG Assertions, Chapter 5.2 - Hard Asserts, and Chapter 5.3 - Soft Asserts.

In this Chapter 5.1, Introduction To TestNG Assertions. We will discuss What Are TestNG Assertions, View The TestNG Assertion Methods, and see why JUnit Assertions and TestNG Assertions are similar to each other. Before discussing assertions, let's go to Eclipse and see why we need assertions.

We will start by running the same Test Script from Chapter 3.3 which setup our test by opening Chrome and the OrangeHRM application, sign in, search for a user, sign out, then tear down our test by closing Chrome and the application. Let's Run.

All 3 Tests Passed but 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. Next in Chapter 5.2, we will cover Hard Asserts.

Chapter 5.2 - Hard Asserts

Hard Asserts

Welcome To Chapter 5.2, Hard Asserts. In the previous Chapter 5.1, we ran our Test Script and it showed Passed without a verification step. The Test Script showed Pass although there is no code in this userSearch Test Method. I'm going to add the code back.

Let's pretend our Test Method requires us to verify the Home Page for OrangeHRM. Go to OrangeHRM and walkthrough the steps. Sign In – Admin / admin123 then click the Login Button. Here's the Home Page. We are going to verify the Welcome hyperlink, Admin tab, and Dashboard. Go back to Eclipse and change the step numbers for userSearch, userSignOut, and teardown. Search User will be 6, User Sign Out will be 7, and Number 8 will be teardown Close Chrome and Application.

Let's add a new Test Method between signIn and userSearch then add some assertions. @Test public void testHomePageVerification. Assert is located in the TestNG class so we write Assert dot and we see the same assertions from Chapter 5.1. Our assertion will be assertEquals.

There're so many methods but they are overloaded. Select an assertion with a String message. We see actual, expected, and message. Actual is true and Expected is true. This assertion will pass because both parameters are equal. The message only shows up if there is a failure so let's write the message like there is an assertion failure. The Welcome Link Is Not Correct On The Home Page. Print statement 3 Verify Welcome Link.

Verify Admin tab. Assert.assertFalse. The description shows Asserts that a condition is false. If it is not an an AssertionError, with the given message, is thrown. Pass this assertion by writing false with a message that states The Admin Tab Is Not Displayed On The Home Page. In order for this assertion to pass the condition must return false. sysout Verify Admin Tab

Verify Dashboard. Assert.assertTrue. Pass this assertion by writing true with a message that states The Dashboard Is Not Correct On The Home Page. sysout 5 Verify Dashboard. This test will Pass. Let's Run. We see all 4 Test Methods Passed. All of the Steps 1 – 8 are printed on the Console including Number 3 Verify Welcome Link, Number 4 Verify Admin Tab, and Number 5 Verify Dashboard. The Results tab also shows each Test Method Passed.

Now, let's see what happens when this Test Method Fails. I'm going to Fail assertEquals for the Welcome Link and assertTrue for the Dashboard. Make the expected result false for assertEquals and the condition false for assertTrue then Run. As expected, testHomePageVerification FAILED.

Let's look at the steps. Steps 3, 4, and 5 are missing. The AssertionError shows our message because our Test failed. The Welcome Link Is Not Correct On The Home Page expected false but found true. Scroll to the bottom and we see no more AssertionErrors. Why is there only 1 AssertionError when 2 assertions failed? The Results tab also shows a Failure with 1 AssertionError.

There is only 1 AssertionError because we used a Hard Assert. A Hard Assert stops immediately after a Failure then move on to the next annotation. In this case, assertEquals statement failed and skipped the subsequent assertions: assertFalse and assertTrue then moved on to the Test Method userSearch which has an @Test annotation. That's not good and that's why we did not see a Print Statement for Steps 3, 4, and 5. For this reason, TestNG introduced Soft Asserts.

Soft Alerts are located in the asserts package and there's only 2 methods: assertAll and doAssert. However, the Soft Assert class can extend the Assertion class so it can use all of these methods.

Next in Chapter 5.3, we will cover Soft Asserts.

Chapter 5.3 - Soft Asserts

Introduction

Welcome To Chapter 5.3, Soft Asserts. In this chapter, we will discuss the Difference Between Hard Asserts & Soft Asserts and answer the question Should Automation Engineers Use Hard and/or Soft Asserts.

Difference Between Hard Asserts & Soft Asserts

The difference between Hard Asserts and Soft Asserts is a Hard Assert stops executions after a fail and move to the next annotation. It does not matter if the next annotation is a Test Annotation or a Configuration Annotation.

We saw in the previous chapter 5.2 steps 3, 4, and 5 did not execute because they were inside the same annotation that failed. Therefore, those steps were skipped and execution picked up at the next Test Method which starts at step 6.

Someone might consider using a try-catch block for the Hard Assert but it won't work either. I'm going to show you. Here's the try-catch block which will try the assertions and catch the AssertionError. Let's Run. Look all of the Test Methods Passed although the AssertionError states The Welcome Link Is Not Correct On The Home Page expected false but found true. Scroll the console. Steps 3, 4, and 5 were skipped again. The Results tab shows all Test Methods Passed. That's not accurate.

A Soft Assert is different from a Hard Assert. It continues execution after a failed assertion and moves to the next statement line. It was designed to keep executing even when a verification step fails.

To use Soft Assert, first, we declare SoftAssert which is a class in TestNG then our object reference softassert equals new SoftAssert. Next, we replace all of the Asserts with our object reference softassert. The assertion methods (assertEquals, assertFalse, and assertTrue) will stay the same.

Recall from the previous video, that the softassert class had only 2 methods: assertAll and doAssert. We need to use assertAll every time for softassert to work. If assertAll is not used then our test will pass and

throw no AssertionError. I'm going to run without assertAll then run with assertAll to show you the difference. Run.

All Test Methods Passed and we know Steps 3 and 5 did not Pass. SoftAssert kept executing although the verification step Failed. Now, let's add assertAll to the end of the Test Method: softassert.assertAll and Run. One of our Test Methods Failed which is testHomePageVerification. Do you see both Assertions? The Welcome Link Is Not Correct On The Home Page expected false but found true. The Dashboard Is Not Correct On The Home Page expected true but found false. All 8 steps show up. The Results tab shows 1 Test Method Failed with the same 2 messages. The Welcome Link Is Not Correct On The Home Page expected false but found true. The Dashboard Is Not Correct On The Home Page expected true but found false. The Welcome Link and Dashboard are not correct on the Home page.

Here's a diagram that shows assertAll. On the left, we see a few assertion methods and assertAll on the right. The arrows are pointing to assertAll because they indicate an AssertionError gets stored into assertAll if there is a failure. That's why we place the assertAll method at the end of our Test Method.

Hard Asserts vs Soft Asserts

Hard Asserts versus Soft Asserts. Which one should we use for automation? We should use both asserts but it depends on our test. Sometimes it's good to have a hard assertion and other times it's not good to have a hard assertion. The same with soft assertions. Let's consider these print statements.

Open Browser, Open Application, Sign Into The Application, Go To Home Page and Verify Home Page, Go To Search Page and Verify Search Page, Search For User, and Sign Out of the Application. We know Hard Asserts stop if a verification step fails and will not execute the next statement but Soft Assert will execute the next statement.

If we had a failure after opening the browser then we would not want to continue executing the next step. There is no reason. With that scenario, it's best to implement a Hard Assert. The same with opening an application. There is no reason to keep executing our Test Script if it's a failure opening the application. The next step which is Sign Into The Application would return a failure.

However, we would not implement a Hard Assert after Step 4, Go To Home Page and Verify Home Page. If there's a failure on the Home Page, we still want to verify Step 5, Go To Search Page and Verify Search Page. To sum up assertions, we want to implement Hard Asserts in our code where it does not make sense to keep executing our test after a verification failure. We implement Soft Asserts in our code when there is a failure and we want to continue executing our test. Next in Chapter 6, we will cover Dependency Testing.