Assertion in selenium WebDriver using TestNG

The word **Assert** means to state a fact or belief confidently or forcefully. In **Selenium**, **Asserts** are validations or checkpoints for an application. **Assertions** state confidently that application behavior is working as expected. One can say that **Asserts in Selenium** are used to validate the test cases.

Assertion determines the state of the application whether it is the same what we are expecting or not. If the assertion fails, then the test case is failed and stops the execution.

To use the Assertion in Web Driver, you need to download the TestNG jar file and add it to the eclipse. Download the TestNG jar file from the link given below:

https://mvnrepository.com/artifact/org.testng/testng/6.7

There are two types of Assertion:

Hard Assertion

Hard Assertion is an Assertion that throws the **AssertException** when the test case is failed. In the case of Hard Assertion, you can handle the error by using a catch block like a java exception. Suppose we have two test cases in a suite. The first test case in a suite has an assertion that fails, and if we want to run the second case in a suit, then we need to handle the assertion error. A Hard Assertion contains the following methods:

- AssertEquals
- AssertNotEquals
- AssertTrue
- AssertFalse
- AssertNull
- AssertNotNull

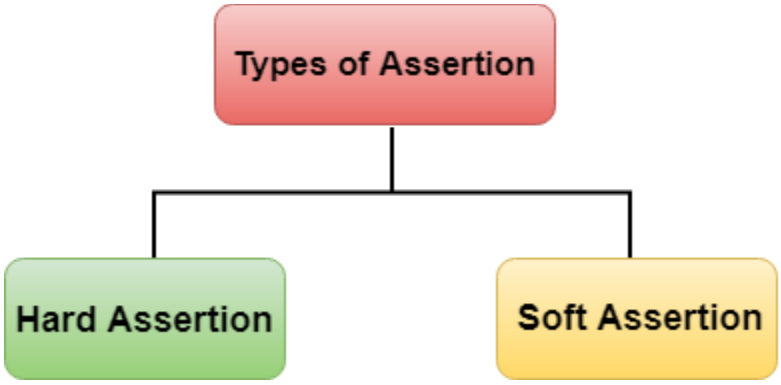
A Selenese tells Selenium what to do. Selenium commands (Selenese) are of three types : Actions, Accessors, and Assertions.

Actions generally manipulate the state of the application like “click this link” and “select that option”. If an Action fails, or has an error, the execution of the current test is stops.

Accessors examine the state of the application and store the results in variables, e.g. “Title”.

Assertions verify that the state of the application is same to what we are expecting.
Selenium Assertions can be of three types: “assert”, “verify”, and ” waitFor”.

When an “assert” fails, the test is aborted.
When a “verify” fails, the test will continue execution, logging the failure.
A “waitFor” command waits for some condition to become true. They will fail and halt the test if the condition does not become true within the current timeout setting. Perhaps, they will succeed immediately if the condition is already true.
When, we talk about the assertions used in WebDriver using TestNg framework, we have two types of assertions; **hard assertion and soft assertion**. Lets see about them briefly:



Hard Assertion in Webdriver using TestNG

A hard assert throw AssertException immediately after a test fails and the test is marked as failed. Perhaps test suite continues with next @Test annotation.

A hard assertion can be of following types:

- assertEquals
- assertNotEquals
- assertTrue
- assertFalse
- assertNull
- assertNotNull

assertEquals with TestNg

This is used to compare expected and actual values in selenium webdriver. Whenever the expected and actual values are same, the assertion passes with no exception. But, if the actual and expected values are not just same, the assert fails with an exception and the test is marked as "failed". The suite continues to run with the next @Test annotation(if any).
Assert.assertEquals(actual, expected);

assertNotEquals

assertNotEquals is just opposite to the functioning of assertEquals assertion. Whenever the expected and actual values matches, the assertion fails with an exception and marks the test-case as "failed". The particular testcase is aborted and execution continuous with the next @Test annotation.
Assert.assertNotEquals(actual, expected, Message);

assertTrue

When we are dealing with Boolean conditions, we should use assertTrue. This assertion returns true if the applied condition passes. If the condition is false/fails, this assertion skips the current method from execution.
Assert.assertTrue(condition);

assertFalse

Assert.assertFalse checks the Boolean value returned by a condition is false or not. When a condition value is true, the assertion aborts the method by an exception. This is basically opposite to assertTrue. The syntax is given below:
Assert.assertFalse(condition);

assertNull

This assertion checks for a object, if it is null or not. When an object is 'null' the assertion returns an exception resulting in aborting the test. The syntax is as follows:
Assert.assertNull(object);

assertNotNull

Assert.assertNotNull is vice-versa of assertNull. When a object has some value, the assertion aborts the method with an exception. The syntax is given below:
Assert.assertNotNull(object);

Now, let's see a simple program which gives a brief idea to use assertion in selenium script.

Soft Assertion In WebDriver Using TestNg

Till now, we have learnt about the hard assertions in WebDriver using TestNg framework. In a hard assertion, when the assertion fails it terminates/aborts the test(method). But, what if we want to run the entire program? What if, we want to report fail test in TestNG report but do not want to terminate the script at any case? It is not really possible if we use hard assertions. So, to overcome this drawback of hard assertion we can use soft assertions in TestNg.
To use a soft assertion in TestNG, we have to include it's corresponding class (as SoftAssert()) in the script. This class prevents the execution to throw any exception (of assertion). Also, the most important context is, now the failed assertions will be reported in the TestNg report and not making the test to abort anywhere.

Let's see an example of soft assertion in TestNG. Also, try to look at the difference between the two assertion and which assert should be used when.

Hard Assertions

```
package AssertionsDemo;

import org.testng.Assert;
import org.testng.annotations.Test;

public class HardAssertDemo {
	@Test
	public void assertEquals() {
		Assert.assertEquals("This assertion will pass", "This assertion will pass");
		System.out.println("This line is executed because assertEquals "
		+ "passed since both the strings are same");

		Assert.assertEquals("assertion", "This assertion will fail");
		System.out.println(
				"This line will not be executed because "
			+ "assertEquals fails both the strings are different."
					+ "Also the test/method will be declared failed");
	}

	@Test
	public void assertNotEquals() {
		Assert.assertNotEquals("This assertion will pass", "Since the "
		  + "expected and actual result do not match");
		System.out.println("This line is executed because assertNotEquals"
```

```
                + " assertion pass for the given situation");
        }

        @Test
        public void assertTrue() {
                Assert.assertTrue(3 < 5);
                System.out.println(
                                "This line will be executed as assertTrue will"
                + " pass because the 3<5(which will return true)");
        }

        @Test
        public void assertFalse() {
                Assert.assertFalse(3 > 5);
                System.out.println("This line is executed because assertFalse"
                                + "assertion passes as the given condition will return false");
        }

        @Test
        public void assertNull() {
                Assert.assertNull(null);
                System.out.println("Since we we set null in the condition, the assertion "
                + "assertNull will pass");
        }

        @Test
        public void assertNotNull() {
                Assert.assertNotNull("This assertion will pass because this "
        + "string don't returns a null value");
                System.out.println("This line is executed because assertNotNull have have passed");
        }
}
```

Soft Assertions

```
package AssertionsDemo;

import org.testng.Assert;
import org.testng.annotations.Test;
import org.testng.asserts.SoftAssert;

public class SoftAssertDemo {
        SoftAssert softAssert = new SoftAssert();
  @Test
  public void hardAssertion(){
        Assert.assertEquals("pass","pass");
        System.out.println("This line is executed because assertEquals "
                        + "passed as both the strings are same");

        Assert.assertNull("assertion");
        System.out.println("Since the object under assertion"
```

```
                        + " is not null, the assertion will fail. "
                        + "This line will not be executed");
    }
    @Test
    public void softAssertion(){

        softAssert.assertNull("assertion");
        System.out.println("We are using Soft assertion in this method,"
                        + " so this line of code will also be executed even if "
                        + "the assetion fails.Wherever we want to execute full "
                        + "testcase/method, we should use SoftAssertion");
        softAssert.assertAll();
    }
}
```