# TestNG Assertion in Selenium | Assert Commands

**TestNG assertion in Selenium** is basically blocks of code that are placed in the test cases to check/verify the success of certain conditions in the test script and decide whether the test has failed or passed.

A test will be considered as successful ONLY if the test is completed without throwing any exception.

That is, If conditions are not true i.e do not satisfy, it will stop the further execution of the test and marks it as failed.

Selenium does not provide any assertion. [*TestNG*](https://www.scientecheasy.com/2019/04/testng-tutorial.html/) provides assertions to check a particular condition. Without using assertions, we will not able to decide whether a test case is failed or not.

## Why do we use TestNG Assertion in Selenium?

There are the following reasons to use assertion in selenium. They are:

1. While performing unit testing, Assertion plays an important role. Most of the unit test frameworks provide implementations by using assertions in the test.

2. An assertion in selenium is used to compare the actual result of an application with the expected result.

3. We mainly use assertions in selenium to verify the result or TestNG reports.

## Types of Assertion in TestNG

There are two types of assertions present in the TestNG framework. They are as follows:

1. Hard Assertion
2. Soft Assertion

Let’s understand them one by one.

## Hard Assertion in TestNG

Hard assertion is a type of assertion that does not continue with execution until the assertion condition is met.

That means when assertion condition does not satisfy and fails, hard assertions usually throw an assertion error and the test case is immediately marked as failed.

An example of hard assertion methods are assertEquals, assertNotEquals, assertTrue, assertFalse, assertNull, assertNotNull.

## Soft Assertion in TestNG

Soft assertion is a type of assertion that continues with test execution even if the assertion condition is not met.

That means soft assertions do not throw an exception when the assert statement (assertion condition) fails and would continue with the execution of the next statement.

A soft assertion in selenium is generally used when our test needs multiple assertions to be executed and the user wants all blocks of code to be executed before failing/skipping tests.

## Assert Class in TestNG

TestNG framework supports the assertion of a test by using assert class which is present in org.testng.Assert class in TestNG under Selenium WebDriver.

Assert class extends Object class. The general syntax to declare assert class in TestNG is as follows:

public class Assert

extends java.lang.Object

This assert is also known as Hard assert. All methods present in the assert class are static. So, we can access all the methods using the class name. We do not need to create an object of Assert class.

## Hard Assert Methods/Commands in TestNG

TestNG Assert class defines the following methods/commands. They are as follows:

**1. assertEquals():** This assertEquals() command basically compares the expected result with that of the actual result. It takes two boolean or string arguments and checks whether both are equal or not.

If not, it will fail the test and will throw an AssertionError and terminates the execution of the program at assert equals method. It has the following general form that is given below:

a. public static void assertEquals(boolean actual, boolean expected) // It takes two boolean arguments.

b. public static void assertEquals(String actual, String expected) // It takes two string arguments.

c. public static void assertEquals(String actual, String expected, String message) // It takes three string arguments and checks that both are equal or not. In case both are not equal, this method fails the test with the given message.

d. public static void assertEquals(java.util.Collection actual, java.util.Collection expected, java.lang.String message)

It takes two collection objects and checks that both of the collections contain the same elements and in the same order. If not, this method will fail the test with the given message.

Let’s verify the Google home page title using assertEquals() method of Assert class.

**Scenario to be automated:**

1. Launch the Firefox browser and visit Google home page.  
2. Verify that the home page title is equal to ‘Google’ or not.  
3. If the page title is not matching with the expected title, assertEquals() method will fail the test case. So, we will provide a message to print “Match not found”.

Let’s create a test case and write code for the above scenario.

**Program source code 1:**

package assertionPrograms;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.Assert;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class VerifyTitle

{

// Create WebDriver reference.

WebDriver driver;

@BeforeClass

public void setUp()

{

// Create an object of FirefoxDriver.

driver = new FirefoxDriver();

driver.manage().window().maximize();

String URL = "https://www.google.com";

driver.get(URL);

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

}

@Test

public void verify()

{

String actualTitle = driver.getTitle();

System.out.println(actualTitle);

// Call assertEqals() method to verify the comparison between actual title and expected title.

Assert.assertEquals(actualTitle, "Google");

System.out.println("Matched Home page title");

}

@Test

public void verify2()

{

String actualTitle = driver.getTitle();

// Call assertEquals() method to verify the comparison between actual title and expected title.

Assert.assertEquals(actualTitle, "Gooooooogle", "Match not found");

System.out.println("Title not matched");

}

@AfterMethod

public void close()

{

driver.close();

}

}

Output:

Google

Matched Home page title

PASSED: verify

FAILED: verify2 java.lang.AssertionError: Match not found expected [Gooooooogle] but found [Google].

In the preceding example program, you can see the first test method verify(), assertEquals statement has been successfully executed, the next statement is executed, and printed “Matched Home page title” on the console.

But in the second test method verify2(), assertEquals statement is not successfully executed because actual title is not matched with the expected title, and comparison failed.

Therefore, TestNG implicitly thrown an exception AssertionError if the condition is not met. Thus, assertEquals() fails the test case and the next statement has been not executed.

**2. assertTrue():** The assertTrue() command verifies that a specified condition is true or not. It takes one boolean argument and checks that a condition is true.

If the condition is true then assertion passes test cases. If boolean value is false, an assertion error is thrown and assertion aborts the test case.

The syntax of assertTrue() method has two general forms which are as follows:

a. public static void assertTrue(boolean condition)

// Overloaded method of assertTrue()

b. public static void assertTrue(String message, boolean condition) // It accepts one boolean argument and String message. If the condition is not true, an AssertionError, with the given message, is thrown.

Let’s take a simple example program where we will compare two contents using the assertTrue() method of Assert class.

**Program source code 2:**

package assertionPrograms;

import org.testng.Assert;

import org.testng.annotations.Test;

public class AssertTrueExample

{

@Test

public void compareNumeric()

{

Assert.assertTrue(5 > 2);

System.out.println("5 is greater than 2");

}

@Test

public void compareString()

{

Assert.assertTrue("Java".equals("Java"), "Not matched"); // true

Assert.assertTrue("Selenium".equals("Java"), "Not matched"); // false

}

}

Output:

5 is greater than 2

PASSED: compareNumeric

FAILED: compareString java.lang.AssertionError: Not macthed expected [true] but found [false]

**3. assertFalse():** The assertFalse() command is similar to assertTrue() method but opposite in nature. It verifies that the specified condition is false or not. If the condition is false (not satisfy), the assertion passes the test case.

If the condition is true, it will throw an AssertionError and marks it as a failed test case. The syntax of assertFalse() method comes into two flavors:

a. public static void assertTrue(boolean condition)

b. public static void assertTrue(boolean condition, String message) // Overloaded version.

Let’s take a simple example program based on assertFalse () method of Assert class.

**Program source code 3:**

package assertionPrograms;

import org.testng.Assert;

import org.testng.annotations.Test;

public class AssertFalseExample

{

@Test

public void compareNumeric()

{

Assert.assertFalse(5 > 2); // Here, condition is true. So, test case will be marked as failed and the next statement will not be executed.

System.out.println("5 is greater than 2");

}

@Test

public void compareString()

{

Assert.assertFalse("Selenium" == "Java", "Matched"); // Here, condition is false. So, test case will be passed by assertion.

}

}

Output:

PASSED: compareString

FAILED: compareNumeric java.lang.AssertionError: expected [false] but found [true]

4. **fail():** The fail() method of Assert class is used to fail the current test with the given message. We can set the error message and also set the exception which caused the failure.

The general syntax for this method is as follows:

public static void fail(String message)

Let’s create a program where we will fail a test case and see the execution of the next statement happens or not.

**Program source code 4:**

package assertionPrograms;

import org.testng.Assert;

import org.testng.annotations.Test;

public class TestFailExample

{

@Test

public void testcasefail()

{

Assert.fail("test purpose"); // Here, test case will be marked as failed and the next statement will not be executed.

System.out.println("Test case failed");

}

}

Output:

FAILED: testcasefail java.lang.AssertionError: test purpose

**5. assertNull():** The assertNull() method is used to verify that a specified object contains a null value or not. This method takes an object as a parameter.

If the object contains a null value, assertion passes the test case otherwise, it will throw an AssertionError if the provided object does not hold a null value.

The general syntax for assertNull() method of Assert class is as follows:

a. public static void assertNull(Object object)

b. public static void assertNull(Object object, String message) // Overloaded version.

Let’s take an example program based on assertNull() method.

**Program source code 5:**

package assertionPrograms;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.Assert;

import org.testng.annotations.Test;

public class AssertNullExample

{

@Test

public void testCase()

{

WebDriver driver = new FirefoxDriver();

Assert.assertNull(driver);

System.out.println("Test case executed successfully");

}

}

Output:

FAILED: testCase java.lang.AssertionError: expected [null] but found [FirefoxDriver: firefox on WINDOWS

**6. assertNotNull():** The assertNotNull() method is used to check that an object does not contain a null value. If an object contains a valid data value, assertion passes the test case otherwise it will throw an AssertionError.

It marks test case as failed if an object contains a null value. The general syntax is given below:

public static void assertNotNull(Object object)

**Program source code 6:**

package assertionPrograms;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.testng.Assert;

import org.testng.annotations.Test;

public class AssertNotNullExample

{

@Test

public void testcase()

{

WebDriver driver = new FirefoxDriver();

Assert.assertNotNull(driver);

System.out.println("Test case executed successfully");

}

}

Output:

Test case executed successfully

PASSED: testcase

**7. assertSame():** The assertSame() method is used to compare the address of objects. If the address of objects is the same then assertion is passed otherwise, TestNG throws an error AssertionError.

The difference between assertEquals and assertSame is, assertEquals() method compares the object contents whereas assertSame() method compares the address of object.

The general syntax for assertSame() method comes into two flavour that are as follows:

a. static public void assertSame(Object actual, Object expected)

b. static public void assertSame(Object actual, Object expected, String message) // Overloaded version.

**Program source code 7:**

package assertionPrograms;

import org.testng.Assert;

import org.testng.annotations.Test;

public class AssertSameExample

{

@Test(priority = 0)

public void testCaseOne()

{

String str1 = "Hello";

String str2 = "Hello";

Assert.assertSame(str1, str2, "Contents are not Same");

System.out.println("Test case one passed successfully");

}

@Test(priority = 1)

public void testCaseTwo()

{

String s1 = "Java";

String s2 = new String("Java");

Assert.assertEquals(s1, s2, "Contents are not same");

System.out.println("Test case two passed successfully");

}

@Test(priority = 2)

public void testCaseThree()

{

String s1 = "Java";

String s2 = new String("Java");

Assert.assertSame(s1, s2, "Address of both string objects are not same");

System.out.println("Test case three passed successfully");

}

}

Output:

Test case one passed successfully

Test case two passed successfully

PASSED: testCaseOne

PASSED: testCaseTwo

FAILED: testCaseThree java.lang.AssertionError: Address of both string objects are not same expected [Java] but found [Java]

**8. assertNotSame():** The assertNotSame() method is used to check that the addresses of objects are not the same. If both the addresses are the same then TestNG throws an AssertionError. The general syntax is as:

a. static public void assertNotSame(Object actual, Object expected)

b. static public void assertNotSame(Object actual, Object expected, String message)

**Program source code 8:**

public class AssertNotSameExample

{

@Test(priority = 0)

public void testCaseOne()

{

String str1 = "Hello";

String str2 = "Hello";

Assert.assertNotSame(str1, str2, "Contents are not same");

System.out.println("Test case one passed successfully");

}

@Test(priority = 1)

public void testCaseTwo()

{

String s1 = "Java";

String s2 = new String("Java"); Assert.assertNotSame(s1, s2, "Address of both string objects are not same");

System.out.println("Test case two passed successfully");

}

}

Output:

Test case two passed successfully

PASSED: testCaseTwo

FAILED: testCaseOne java.lang.AssertionError: Contents are not Same expected not same [Hello] but found [Hello]

## SoftAssert Class in TestNG

TestNG framework supports the soft assertion of a test by using SoftAssert class which is present in org.testng.Assert class. SoftAssert class extends the Assertion class that extends Object class.

Through inheritance, SoftAssert Class allows users to access all the methods present in the Assertion class.

The general declaration for SoftAssert class is as follows:

public class SoftAssert

extends Assertion

The soft assertion is also known as verify. The methods present SoftAssert class are non-static. So, we need to create an object of SoftAssert class and then we can access all the methods using the object reference variable.

When an assertion fails, SoftAssert doesn’t throw an exception but records the failure and continues with the next execution of statements.

## SoftAssert Methods/Commands in Selenium

In addition to methods inherited from Assertion class, SoftAssert class provides only two new methods. They are as follows:

1. **doAssert (protected method, we would not be using it)**  
2. **assertAll()**

Let’s take an example program based on soft assert and hard assert.

**Program source code 9:**

package assertionPrograms;

import org.testng.Assert;

import org.testng.annotations.Test;

import org.testng.asserts.SoftAssert;

public class SoftAssertExample

{

@Test(priority = 0)

public void softAssert()

{

SoftAssert sa = new SoftAssert();

System.out.println("Soft assertion started from here");

sa.assertTrue(false); // Test case failed here but next statement will be executed.

System.out.println("Soft assertion ended here");

}

@Test(priority = 1)

public void hardAssert()

{

System.out.println("Hard assertion started from here");

Assert.assertTrue(false); // Test case failed here and next statement will not be executed.

System.out.println("Hard assertion ended here");

}

}

Output:

Soft assertion started from here

Soft assertion ended here

Hard assertion started from here

PASSED: softAssert

FAILED: hardAssert java.lang.AssertionError: expected [true] but found [false]

As you can see in the coding, when the test case failed in softAssert, it did not throw any exception error and continue with execution of the next statement.

But in the case of hardAssert, when the test case failed, it has thrown an AssertionError and did not execute the next statement. Look at the output of the program.

**1. assertAll():** The assertAll() method of softAssert class is used to know the failure record when assertion failed. This method cannot be used alone in the test case. It must be used as the last statement along with some other assert methods.

Let’s modify the previous coding to know the behavior of assertAll() method.

**Program source code 10:**

package assertionPrograms;

import org.testng.Assert;

import org.testng.annotations.Test;

import org.testng.asserts.SoftAssert;

public class AssertAllExample

{

@Test(priority = 0)

public void softAssert()

{

SoftAssert sa = new SoftAssert();

System.out.println("Soft assertion started from here");

sa.assertTrue(false);

System.out.println("Soft assertion ended here");

sa.assertAll();

}

@Test(priority = 1)

public void hardAssert()

{

System.out.println("Hard assertion started from here");

Assert.assertTrue(false);

System.out.println("Hard assertion ended here");

}

}

Output:

Soft assertion started from here

Soft assertion ended here

Hard assertion started from here

FAILED: softAssert java.lang.AssertionError: The following asserts failed: expected [true] but found [false]

FAILED: hardAssert java.lang.AssertionError: expected [true] but found [false]

Observe the output and compare it with the previous output.