

assertTrue() method

This lesson demonstrates how to use assertTrue method in JUnit 5 to assert test conditions.

We'll cover the following ^

- assertTrue() method
- Demo
- Explanation -

assertTrue() method

Assertions API provide static `assertTrue()` method. This method helps us in validating that the actual value supplied to it is `true`.

- If the actual value is `true` then test case will pass.
- If the actual value is `not true` then test case will fail.

There are basically six useful overloaded methods for assertTrue:-

```
1 public static void assertTrue(boolean condition)
2
3 public static void assertTrue(boolean condition, String message)
4
5 public static void assertTrue(boolean condition, Supplier<String> messageSupplier)
6
7 public static void assertTrue(BooleanSupplier booleanSupplier)
8
9 public static void assertTrue(BooleanSupplier booleanSupplier, String message)
10
11 public static void assertTrue(BooleanSupplier booleanSupplier, Supplier<String> messageS
```



Java Unit Testing with JUnit 5

JUnit 5 Assertions – assertTrue() method



Dinesh Varyani

<https://www.hubberspot.com>

Demo

Let's look into the usage of the above methods:-

```
package io.educative.junit5;

import static org.junit.jupiter.api.Assertions.assertTrue;

import org.junit.jupiter.api.Test;

public class AssertTrueDemo {

    @Test
    public void testAssertTrueWithTrueCondition() {
        boolean trueValue = true;
        assertTrue(trueValue);
    }

    @Test
    public void testAssertTrueWithFalseCondition() {
        boolean falseValue = false;
        assertTrue(falseValue);
    }

    @Test
    public void testAssertTrueWithFalseConditionAndMessage() {
        boolean falseValue = false;
        assertTrue(falseValue, "The actual value is false");
    }
}
```



```

@Test
public void testAssertTrueWithFalseConditionAndMessageSupplier() {
    boolean falseValue = false;

    assertTrue(falseValue, () -> "The actual value is false");
}

@Test
public void testAssertTrueWithBooleanSupplier() {
    boolean trueValue = true;
    assertTrue(() -> trueValue);
}

@Test
public void testAssertTrueWithBooleanSupplierAndMessage() {
    boolean falseValue = false;
    assertTrue(() -> falseValue, "The actual value is false");
}

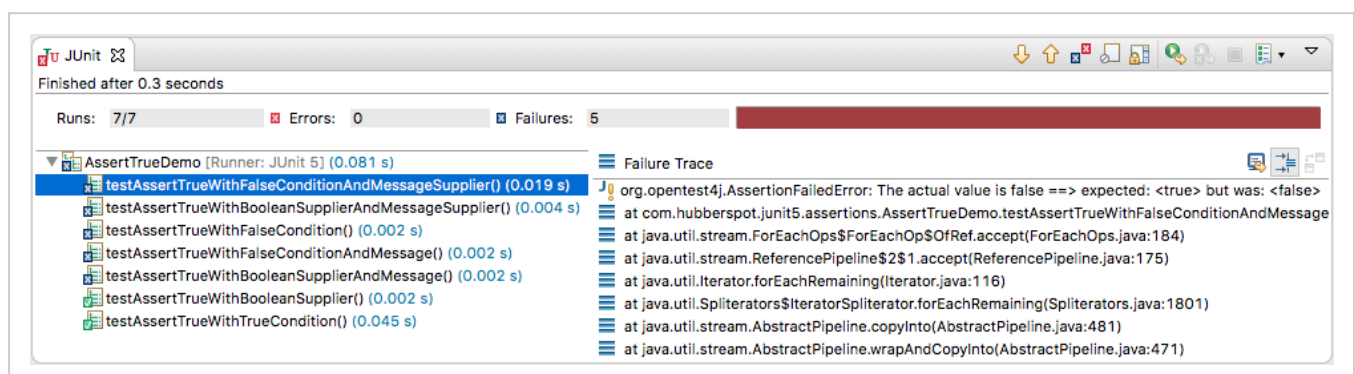
@Test
public void testAssertTrueWithBooleanSupplierAndMessageSupplier() {
    boolean falseValue = false;
    assertTrue(() -> falseValue, () -> "The actual value is false");
}
}

```



You can perform code changes to above code widget, run and practice different outcomes.

Run AssertTrueDemo class as JUnit Test.



Explanation -

In AssertTrueDemo class there are 7 @Test methods. These 7 methods demonstrate the working of the above 6 overloaded methods of **assertTrue** :-

1. **testAssertTrueWithTrueCondition()** - It asserts the boolean value provided to **assertTrue()** method. Here, the actual value passed to it is

`true`. Thus, it passes the JUnit test case because `assertTrue` asserts that value passed to it should be true.

2. `testAssertTrueWithFalseCondition()` - It asserts the boolean value provided to `assertTrue()` method. Here, actual value passed to it is `false`. Thus, it fails the JUnit test case with `AssertionFailedError: expected: <true> but was: <false> because value passed to assertTrue method is false.`
3. `testAssertTrueWithFalseConditionAndMessage` - It asserts the boolean value provided to `assertTrue()` method. Here, actual value passed to it is `false`. Thus, it fails the JUnit test case with `AssertionFailedError: The actual value is false ==> expected: <true> but was: <false> because value passed to assertTrue method is false. It gives AssertionFailedError followed String message we provide to assertTrue() method.`
4. `testAssertTrueWithFalseConditionAndMessageSupplier` - It asserts the boolean value provided to `assertTrue()` method. Here, actual value passed to it is `false`. Thus, it fails the JUnit test case with `AssertionFailedError: The actual value is false ==> expected: <true> but was: <false> because value passed to assertTrue method is false. It gives AssertionFailedError followed by lazily evaluates String message we provide to assertTrue() method, as lambda expression.`
5. `testAssertTrueWithBooleanSupplier()` - It asserts the boolean value provided to `assertTrue()` method through `BooleanSupplier` functional interface. Here, actual value passed to it is `true`. Thus, it passes the JUnit test case because `assertTrue` asserts that value passed to it should be true.
6. `testAssertTrueWithBooleanSupplierAndMessage` - It asserts the boolean value provided to `assertTrue()` method through `BooleanSupplier` functional interface. Here, actual value passed to it is `false`. Thus, it fails the JUnit test case with `AssertionFailedError: The actual value is false ==> expected: <true> but was: <false> because value passed to assertTrue method is false. It gives AssertionFailedError followed String message we provide to assertTrue() method.`

7. `testAssertTrueWithBooleanSupplierAndMessageSupplier` - It asserts the boolean value provided to `assertTrue()` method through `BooleanSupplier` functional interface. Here, actual value passed to it is `false`. Thus, it fails the Junit test case with `AssertionFailedError`: The actual value is false ==> expected: <true> but was: <false> because value passed to `assertTrue` method is `false`. It gives `AssertionFailedError` followed by lazily evaluates `String message` we provide to `assertTrue()` method, as lambda expression.
-

In the next lesson, we will look into `assertFalse()` assertion.