

assertNull() method

This lesson demonstrates how to use `assertNull()` method in JUnit 5 to assert test conditions.

We'll cover the following

- `assertNull()` method
- Demo
- Class Under Test - `StringUtils`
- Output
- Explanation -


assertNull() method

Assertions API provide static `assertNull()` method. This method helps us in validating that the particular object is null. This method takes an actual value and checks whether it is `null` or not.

- If the actual value is **null** then test case will pass.
- If the actual value is **not null** then test case will fail.

There are basically three overloaded methods for `assertNull` which are described below:-


```
1 public static void assertNull(Object actual)
2
3 public static void assertNull(Object actual, String message)
4
5 public static void assertNull(Object actual, Supplier<String> messageSupplier)
6
```



1. `assertNull(Object actual)` - It asserts whether actual value is null or not.
2. `assertNull(Object actual, String message)` - It asserts whether actual value is null or not. In case, if the actual value is not null then the test

case will fail with a provided message.

3. `assertNull(Object actual, Supplier<String> messageSupplier)` - It asserts whether the actual value is null or not. In case, if the actual value is not null, then the test case will fail with a provided message through the Supplier function. The main advantage of using the Supplier function is that it lazily evaluates to String only when the test case fails.

 JUnit 5 Assertions - assertNull method

Java Unit Testing with JUnit 5

JUnit 5 Assertions – assertNull() method

Dinesh Varyani
<https://www.hubberspot.com>

assertNull method

Demo

Step 1 - Create a Java class in Eclipse as discussed in previous lessons.

Step 2 - Give it a name as, StringUtils.

StringUtils.java

```
package com.hubberspot.junit5.assertions;

public class StringUtils {

    public static String reverse(String input) {
        if(input == null) {
            return null;
        }
    }
}
```

```

        if(input.length() == 0) {
            return "";
        }

        char[] charArray = input.toCharArray();
        int start = 0;
        int end = input.length() - 1;

        while(start < end) {
            char temp = charArray[start];
            charArray[start] = charArray[end];
            charArray[end] = temp;
            start++;
            end--;
        }

        return new String(charArray);
    }
}

```

Class Under Test - StringUtils

StringUtils is our class under test. It has one method as, `reverse()`. This method takes in a String and returns reverse of it.

For example -

1. If we provide input String as, "ABCD", it returns back "DCBA".
2. If we provide input String as, "Student", it returns back "tneduts".
3. If we provide input String as, **null**, it returns back **null**.
4. If we provide input String as, "", it returns back "" String.

Step 3 - Create a test class by name, `"StringUtilsTest"`. This test class will demonstrate all overloaded `assertNull()` methods.

StringUtilsTest.java

StringUtils.java



```

package io.educative.junit5;

import static org.junit.jupiter.api.Assertions.*;
import java.util.function.Supplier;
import org.junit.jupiter.api.Test;

class StringUtilsTest {

```

```
// ***** assertNull Example - Start *****
@Test
void givenNullString_whenReverseIsCalled_thenNullIsReturned() {
    String actual = StringUtils.reverse((null));

    // assertNull without message
    assertNull(actual);
}

@Test
void givenEmptyString_whenReverseIsCalled_thenEmptyStringIsReturned() {
    String actual = StringUtils.reverse("");
    String message = "Actual String should be null !!! ";

    // assertNull with message
    assertNull(actual, message);
}

@Test
void givenNonNullString_whenReverseIsCalled_thenReversedStringIsReturned() {
    String actual = StringUtils.reverse("ABCD");
    Supplier<String> messageSupplier = () -> "Actual String should be null !!! ";

    // assertNull with Java 8 MessageSupplier
    assertNull(actual, messageSupplier);
}

// ***** assertNull Example - End *****
}
```



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

Step 4 - Run StringUtilsTest class as Junit Test.

Output

JUnit 5 [Runner: JUnit 5] (0.001 s)

Finished after 0.146 seconds

Runs: 3/3 Errors: 0 Failures: 2

Failure Trace

```
org.opentest4j.AssertionFailedError: Actual String should be null !!! ==> expected: <null> but was: <DCBA>
    at com.hubberspot.junit5.assertions.StringUtilsTest.givenNonNullString_whenReverseIsCalled_thenReverse...
    at java.util.stream.ForEachOps$ForEachOp$OfRef.accept(ForEachOps.java:184)
    at java.util.stream.ReferencePipeline$2$1.accept(ReferencePipeline.java:175)
    at java.util.Iterator.forEachRemaining(Iterator.java:116)
    at java.util.Spliterators$IteratorSpliterator.forEachRemaining(Spliterators.java:1801)
    at java.util.stream.AbstractPipeline.copyInto(AbstractPipeline.java:481)
    at java.util.stream.AbstractPipeline.wrapAndCopyInto(AbstractPipeline.java:471)
    at java.util.stream.ForEachOps$ForEachOp.evaluateSequential(ForEachOps.java:151)
    at java.util.stream.ForEachOps$ForEachOp$OfRef.evaluateSequential(ForEachOps.java:174)
    at java.util.stream.AbstractPipeline.evaluate(AbstractPipeline.java:234)
    at java.util.stream.ReferencePipeline.forEach(ReferencePipeline.java:418)
    at java.util.stream.ForEachOps$ForEachOp$OfRef.accept(ForEachOps.java:184)
    at java.util.stream.ReferencePipeline$2$1.accept(ReferencePipeline.java:175)
    at java.util.Iterator.forEachRemaining(Iterator.java:116)
    at java.util.Spliterators$IteratorSpliterator.forEachRemaining(Spliterators.java:1801)
    at java.util.stream.AbstractPipeline.copyInto(AbstractPipeline.java:481)
```

Explanation -

The order of execution of test cases depends on Junit 5. In StringUtilsTest class there are 3 @Test methods:-

1. `givenNullString_whenReverseIsCalled_thenNullIsReturned()` - It tests the scenario that when **null** is provided to `reverse()` method of StringUtils class, then **null** is returned. So, on **line 15** providing `assertNull()` asserts that actual value returned is null. Thus, it passes the Junit test case.
2. `givenEmptyString_whenReverseIsCalled_thenEmptyStringIsReturned()` - It tests the scenario that when "" is provided to `reverse()` method of StringUtils class, then "" is returned. Here, return value is empty string which is not null. So, on **line 24** providing `assertNull()` asserts that actual value returned is null. Thus, it fails the Junit test case because actual value returned is "". In this test case, we are using overloaded `assertNull()` method, which takes **String message** as second argument. As, this test case doesn't satisfy assertion condition, it fails and give "`AssertionFailedError`: Actual String should be null !!! ==> expected: but was: <>". It gives `AssertionFailedError` followed by **String message** we provide to `assertNull()` method.
3. `givenNonNullString_whenReverseIsCalled_thenReversedStringIsReturned` - It tests the scenario that when **ABCD** is provided to `reverse()` method of StringUtils class, then **DCBA** is returned. Here, return value is not null. So, on **line 33** providing `assertNull()` asserts that actual value returned is null. Thus, it fails the Junit test case because actual value returned is **DCBA**. In this test case, we are using overloaded `assertNull()` method, which takes `Supplier<String> messageSupplier` as second argument. As, this test case doesn't satisfy assertion condition, it fails and give "`AssertionFailedError`: Actual String should be null !!! ==> expected: but was: <DCBA>". It gives `AssertionFailedError` followed by lazily evaluates **String message** we provide to `assertNull()` method, as lambda expression.

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

