How to assert that a certain exception is thrown in jUnit4.5 tests

51

15

How can use jUnit4.5 idiomatically to test that come code throws an exception?

While I can certainly do something like this:

```
@Test
public void testFooThrowsIndexOutOfBoundsException() {
  boolean thrown = false;

  try {
    foo.doStuff();
  } catch (IndexOutOfBoundsException e) {
    thrown = true;
  }

  assertTrue(thrown);
}
```

I recall that there is an annotation or an Assert.xyz or *something* that is far less cludgy and far more in-the-spirit of jUnit for these sorts of situations.

java junit

link | improve this question

asked Oct 1 '08 at 6:56
SCdF
6,924 ●5 ●26 ●66
84% accept rate

feedback

6 Answers

active oldest votes

55

JUnit 4 has support for this:

```
@Test(expected=IndexOutOfBoundsException.class)
public void testIndexOutOfBoundsException() {
    ArrayList emptyList = new ArrayList();
    Object o = emptyList.get(0);
}
```

link | improve this answer

answered Oct 1 '08 at 7:12
skaffman
115k ●8 ●136 ●227

3 I discovered that this test doesn't fail if you get no exception. So the expected is more like an possible assumption and not a mandatory one. So in this case, your test declare that there might be thrown an IndexOutOfBoundsException, and not that the test should thrown such exception. – raisercostin Dec 10 '09 at 14:48

Hello World!

This is a collaboratively edited question and answer site for professional and enthusiast programmers. It's 100% free, no registration required.

about » faq »

tagged

java × 234102

junit × 2976

asked 3 years ago
viewed 28,474 times
active 5 months ago

Community Bulletin

Join us for Stack Overflow Worldwide Meetups! – Apr 28

Linked

Cause of an unexpected behaviour using JUnit 4's expected exception mechanism?

Related

In Java how can I validate a thrown exception with JUnit?

How can I make my JUnit tests run in random order?

Re-running failed and not-run tests

How can I run all junit tests in the project by using console?

How to run all tests belonging

2 I discovered that that what i described before happened because the test wasn't marked with this annotation <code>@RunWith(value=BlockJUnit4ClassRunner.class)</code> – raisercostin Dec 10 '09 at 15:08

This piece of code will not work if you expect an exception only somewhere in your code, and not a blanket like this one. – Chin Boon Jun 27 '11 at 14:50

feedback

76

If you can use JUnit 4.7, you can use the `ExpectedException` Rule

```
@RunWith(JUnit4.class)
public class FooTest {
  @Rule
  public ExpectedException exception = ExpectedException.none();

  @Test
  public void doStuffThrowsIndexOutOfBoundsException() {
    Foo foo = new Foo();

    exception.expect(IndexOutOfBoundsException.class);
    foo.doStuff();
  }
}
```

This is much better than `@Test(expected=IndexOutOfBoundsException.class)` because the test will fail if `IndexOutOfBoundsException` is thrown before `foo.doStuff()`

See this article for details

link | improve this answer

answered May 29 '10 at 17:16
NamshubWriter
2,795 ●5 ●14

4 +1 a better solution than the accepted answer. – Chin Boon Jun 27 '11 at 14:51

feedback

30

Be careful using expected exception, because it only asserts that the method threw that exception, not a particular line of code in the test.

I tend to use this for testing parameter validation, because such methods are usually very simple, but more complex tests might better be served with:

```
try {
    methodThatShouldThrow();
    fail( "My method didn't throw when I expected it to" );
} catch (MyException expectedException) {
}
```

Apply judgement.

link | improve this answer

edited Oct 1 '08 at 9:36

answered Oct 1 '08 at 9:31
daveb
7,967 ●1 ●8 ●20

to a certain Category in JUnit 4
Running JUnit integration tests from a JSP?
junit assert in thread throws exception
Assert List in Junit
Is there a tool for Java which finds which lines of code are tested by specific JUnit tests?
Assert statement causing JUnit tests to stop
How to prepare state for several JUnit tests only once
How to ensure quality of junit tests?
How to group/categorize large number of JUnit tests
How to make log4j error() calls throw an exception in jUnit tests?
Organizing JUnit tests
How can I test if a particular exception is not thrown?
Custom Junit runner doesn't ignore tests with Assume.assumeTrue(someFalse
How can I make JUnit let me set variables in one test case and access them in other if they are in the same class
JUnit, Timeout for parameterized tests
A unit test is intended to check for a particular failure, but it still fails even though the expected exception is thrown?
Does tearDown get called if test case throws exception? (JUnit)
How would you test something with complex output?
In a Java unit test, how do I assert a number is within a given range?
In Java unit tests, is it possible to include an assert statement within a loop?
How to access the fields of a test class with in an Rule in JUnit

3 Maybe I'm old school but I still prefer this. It also gives me a place to test the exception itself: sometimes I have exceptions with getters for certain values, or I might simply look for a particular value in the message (e.g. looking for "xyz" in the message "unrecognized code 'xyz'"). – Rodney Gitzel Oct 6 '10 at 17:22

I think NamshubWriter's approach gives you the best of both worlds. – Eddie Mar 9 '11 at 19:21

+1 useful in some scenarios where expected = xx doesn't match requirements. – Chin Boon Jun 27 '11 at 15:20

feedback

5

To solve the same problem I did set up a small project: http://code.google.com/p/catch-exception/

Using this little helper you would write

```
verifyException(foo, IndexOutOfBoundsException.class).doStuff();
```

This is less verbose than the ExpectedException rule of JUnit 4.7. In comparison to the solution provided by skaffman, you can specify in which line of code you expect the exception. I hope this helps.

link | improve this answer

answered Oct 28 '11 at 9:26
user998938
51 ●1 ●1

feedback

2

How about this: Catch a very general exception, make sure it makes it out of the catch block, then assert that the class of the exception is what you expect it to be. This assert will fail if a) the exception is of the wrong type (eg. if you got a Null Pointer instead) and b) the exception wasn't ever thrown.

```
public void testFooThrowsIndexOutOfBoundsException() {
  Throwable e = null;

  try {
    foo.doStuff();
  } catch (Throwable ex) {
    e = ex;
  }

  assertTrue(ex instanceof IndexOutOfBoundsException);
}
```

link | improve this answer

answered Oct 1 '08 at 7:03
Johan
836 ●4 ●12

1 That's what you do in JUnit 3. Junit 4 does it better. – skaffman Oct 1 '08 at 7:24

feedback

2

JUnit has built-in support for this, with an "expected" attribute

link | improve this answer

answered Oct 1 '08 at 7:05
Mark Bessey
10.9k ●3 ●16 ●32

feedback

Your Answer

log in

or

Name

Email

Home Page

Post Your Answer

Not the answer you're looking for? Browse other questions tagged java junit or ask your own question.

question feed