UNIVERSIDADE FEDERAL DE SERGIPE

VINÍCIUS DIAS VALENÇA

Teste DE SOFTWARE 2
ATIVIDADE 1
Problema do Stack Overflow

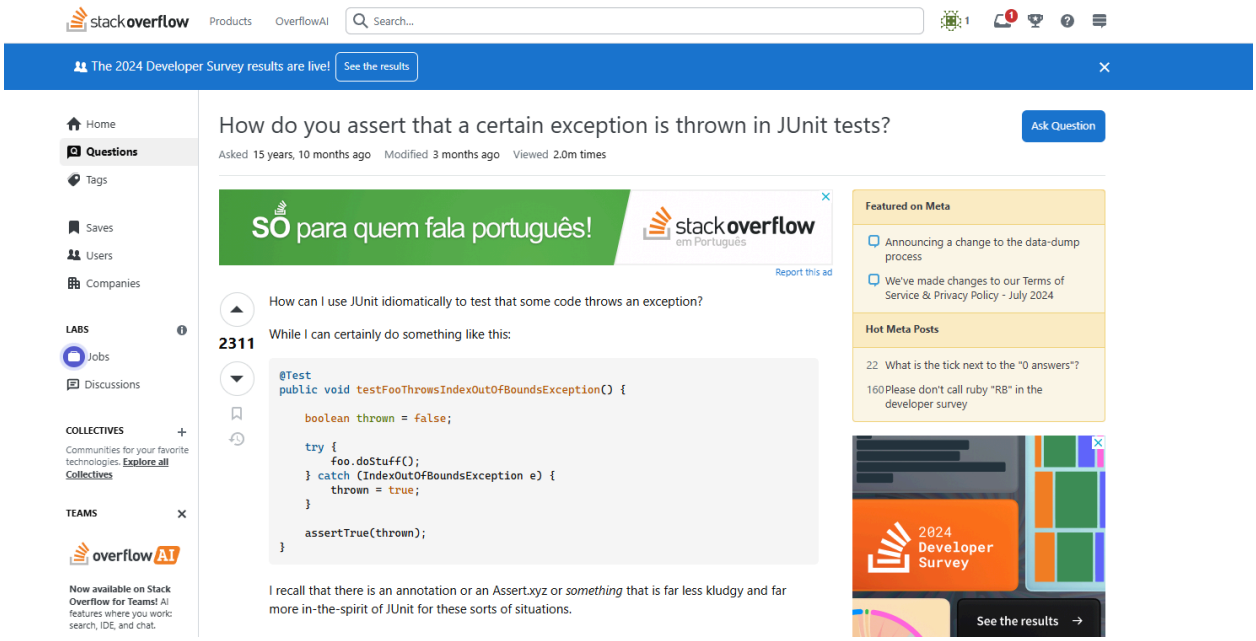
São Cristóvão

2024

1. Motivação

How do you assert that a certain exception is thrown in JUnit tests?

https://stackoverflow.com/questions/156503/how-do-you-assert-that-a-certain-exception-is-thrown-in-junit-tests




```
@Test
public void testFooThrowsIndexOutOfBoundsException() {

    boolean thrown = false;

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

    assertTrue(thrown);
}
```

Ao escrever testes unitários, um cenário comum é garantir que uma exceção específica seja lançada quando certas condições são atendidas. Por exemplo, se um método deve lançar uma exceção quando recebe parâmetros inválidos, queremos testar e garantir esse comportamento.

Problemas com a Abordagem Inicial

1. A necessidade de um bloco `try-catch` e a variável `thrown` adicionam código desnecessário, tornando o teste mais verboso e difícil de ler.
2. O teste depende da variável `thrown` ser corretamente definida no bloco `catch`. Qualquer alteração acidental pode fazer o teste falhar silenciosamente.
3. Se o teste falhar, a mensagem de erro padrão pode não ser muito clara sobre o que deu errado.

2. Reproduzindo o problema

```java
package com.example;
import java.util.List;

public class ClassToTest {
    public void checkNames(List<String> names) {
        for (String name : names) {
            if (name.length() <= 2) {
                throw new IllegalArgumentException("O nome precisa ter pelo menos 3 letras: " + name);
            }
        }
    }
}
```

```java
package com.example;
import org.junit.jupiter.api.Test;

import java.util.Arrays;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertThrows;

public class TestClass {

    @Test
    public void testCheckNamesThrowsExceptionForShortNames() {
        ClassToTest foo = new ClassToTest();
        List<String> names = Arrays.asList(...a:"Joe", "Al", "Sam");


        IllegalArgumentException exception = assertThrows(expectedType:IllegalArgumentException.class, () -> foo.checkNames(names));
        assertEquals(expected:"O nome precisa ter pelo menos 3 letras: Al", exception.getMessage());
    }

    @Test
    public void testCheckNamesDoesNotThrowExceptionForValidNames() {
        ClassToTest foo = new ClassToTest();
        List<String> names = Arrays.asList(...a:"Joe", "Sam", "Ann");


        foo.checkNames(names);
    }
}
```

`testCheckNamesThrowsExceptionForShortNames`:

Cria uma instância de `ClassToTest` e uma lista de nomes onde um dos nomes é muito curto.

Usa `assertThrows` para verificar que `IllegalArgumentException` é lançada.

Verifica se a mensagem da exceção corresponde à esperada.

`testCheckNamesDoesNotThrowExceptionForValidNames`:

Cria uma instância de `ClassToTest` e uma lista de nomes válidos.

Chama `checkNames` e verifica implicitamente que nenhuma exceção é lançada.

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>atividade1-project</artifactId>
    <version>1.0-SNAPSHOT</version>

    <dependencies>
        <dependency>+
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>5.8.2</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>5.8.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>3.0.0-M5</version>
            </plugin>
        </plugins>
    </build>
</project>
```

3. Outras soluções:

in junit, there are four ways to test exception.

248

junit5.x

- for junit5.x, you can use `assertThrows` as following

```
@Test
public void testFooThrowsIndexOutOfBoundsException() {
    Throwable exception = assertThrows(IndexOutOfBoundsException.class, () -> foo
    assertEquals("expected messages", exception.getMessage());
}
```

Utilizar uma exceção específica, como `IllegalArgumentException`, comunica claramente a intenção do código e o tipo de erro que está sendo tratado. Isso torna o código mais legível e fácil de entender para outros desenvolvedores que possam trabalhar nele no futuro.

`Throwable` é a superclasse de todas as exceções e erros em Java. Utilizá-la para capturar exceções pode levar a capturar exceções não intencionais, como `NullPointerException`, `IOException`, ou mesmo `OutOfMemoryError`, que são casos que normalmente não deveriam ser tratados da mesma forma.