UNIVERSIDADE FEDERAL DE SERGIPE

LUAN FABRICIO DE CARVALHO

ATIVIDADE 01

UNIDADE 01

TESTE DE SOFTWARE

SÃO CRISTÓVÃO

2024

Justificativa para o uso da resposta

A seguinte resposta foi utilizada como base:

967

You can use `assertThrows()`, which allows you to test multiple exceptions within the same test. With support for lambdas in Java 8, this is the canonical way to test for exceptions in JUnit.

Per the JUnit docs:

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

@Test
void exceptionTesting() {
    MyException thrown = assertThrows(
           MyException.class,
           () -> myObject.doThing(),
           "Expected doThing() to throw, but it didn't"
    );

    assertTrue(thrown.getMessage().contains("Stuff"));
}
```

+200

Share Follow

edited Jun 4, 2023 at 15:22

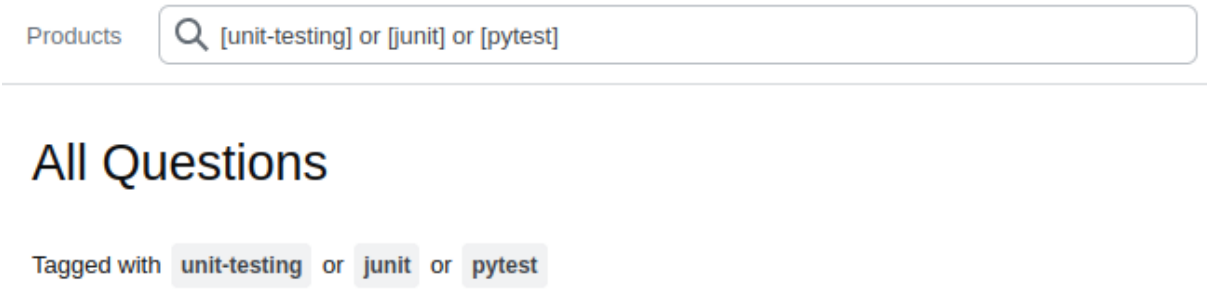
community wiki
14 revs, 9 users 58%
steventrouble

Ela foi utilizada, pois apresenta um texto bem curto apresentando a solução e referenciando o código-fonte e documentação oficial do JUnit, e no fim mostra um exemplo bem simples e prático para facilitar o entendimento.

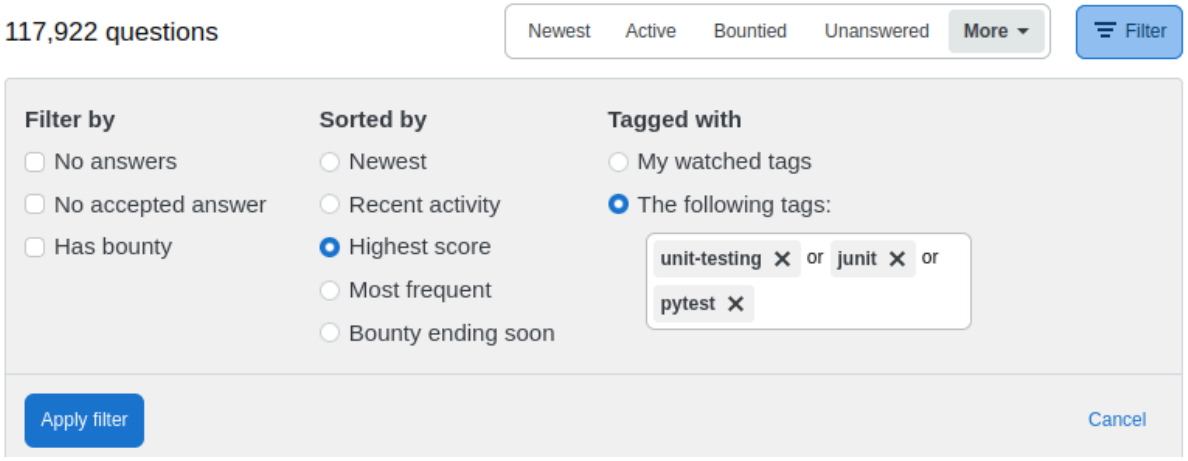
A maioria das outras respostas também estavam corretas, porém, não eram tão claras ou não apresentavam exemplos simples.

Etapa 1

Para esta etapa, foi utilizado o StackOverflow, para escolher uma pergunta referente a teste de software. Nele foram utilizados os seguintes filtros: unit-testing, junit ou pytest, como pode ser visto no print.



Além disso, a opção de ordenar as respostas com base na pontuação, de forma decrescente (highest score).



Com o filtro e ordenações feitas, foi apenas necessário procurar uma pergunta com resposta aceita que possui pelo menos 400 votos e que não seja repetida.

Etapa 2

Pré-requisitos

- Java OpenJDK 17 ou superior instalado.

Escolhendo um projeto base

Como a pergunta se refere a melhor forma de testar Exceptions, foi necessário utilizar um projeto que possua JUnit5 já configurado. Nesse caso foi utilizado o repositório do **JUnit-team**, o junit5-samples, ele possui vários projetos base para diferentes programas de build, onde o programa utilizado para esse exemplo foi utilizado o ant.

Para baixar e acessar o repositório correto, foram utilizados os seguintes comandos.

$ git clone --depth=1 https://github.com/junit-team/junit5-samples.git

Para baixar o repositório e

$ cd junit5-samples/junit5-jupiter-starter-ant

para mudar de diretório.

Configurando o ambiente

Dentro do projeto **junit5-jupiter-starter-ant**, é necessário rodar o script **build.sh**, para baixar as dependências e executar o processo de build. Assim o ambiente já está configurado.

Reproduzindo caso de teste

Nessa pergunta, o autor relata que possui problemas quando espera que vários métodos lancem uma exceção (utilizando o @Rule). Então, será implementado um caso onde um método chama outro que pode lançar uma exceção.

Como o projeto base implementa uma calculadora para demonstrar os testes do JUnit, serão implementados dois métodos neste tutorial, o **div** (para calcular a divisão) e o **avg** (para calcular a média de um array).

Primeiro, será implementado o método **div**, pois ele será utilizado no método **avg**. Ele receberá dois argumentos **a** e **b** e retornará o resultado de **a / b**, caso **b** seja diferente de zero. Caso **b** seja igual a zero, será lançada uma exceção com a mensagem "Divisão por zero". O código do método **div** será o seguinte:

```java
package com.example.project;

public class Calculator {

        public int add(int a, int b) {
                return a + b;
        }

        public float div(float a, float b) throws Exception {
                if (b == 0) throw new Exception("Divisão por zero");
                return a / b;
        }
}
```

Observação: O print é do arquivo **Calculator.java**, dentro da pasta **src/main/java/com/example/project/**.

Após implementar o método **div**, a próxima etapa será implementar o teste desse método. Como o tutorial é focado na implementação de testes para tratar o lançamento de exceção dos métodos, o único teste implementado para esse método será para validar se ele lançou uma exceção.

Antes de escrever os testes, é necessário realizar o import da função **assertThrows**, para verificar se a exceção foi lançada corretamente. Para fazer isso basta adicionar a seguinte linha antes da declaração da classe **CalculatorTests:**

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

Observação: **CalculatorTest** fica dentro do arquivo **src/test/java/com/example/project/CalculatorTest.java**.

Para implementar este teste, basta modificar a classe **CalculatorTests**, adicionando o método **divThrows** com o decorator **@Test**. Dentro desse método, é necessário criar uma instância da classe **Calculator**, depois disso será utilizada a função **assertThrows** para executar o método div e esperar o lançamento de um erro. Para finalizar, basta apenas verificar se a mensagem de erro recebida bate com a mensagem de erro esperada.
O código do método **divThrows** será parecido com o seguinte:

```
@Test
@DisplayName("42 / 0 => throw")
void divThrows() {
        Calculator calc = new Calculator();
        Exception e = assertThrows(
                        Exception.class,
                        () -> calc.div(42.0f, 0.0f),
                        "Expected div(?, 0.0) to throw, but it didn't"
                        );


        assertTrue(e.getMessage().contains("zero"));
}
```

Nesta etapa, o método **avg** será implementado (dentro da classe **Calculator**), ele recebe um array de floats e retorna a média aritmética dos valores desse array. Para calcular a média, ele usa o método **div** então, caso ocorra uma divisão por zero, o método também deve lançar uma exceção. O código desse método é o seguinte:

```
public float avg(float[] values) throws Exception {
        float sum = 0;
        float len = values.length;

        for (int i = 0; i < len; ++i) {
                sum += values[i];
        }

        return div(sum, len);
}
```

A implementação do teste para o método **avg** (na classe **CalculatorTest**) é muito parecida com a do método **div**. Nele é necessário instanciar a classe **Calculator**, iniciar uma variável que possui um array vazio e, utilizar os métodos **assertThrows** e **assertTrue** para verificar se o método lançou uma exceção e se a mensagem da exceção é a esperada. Como no print:

```java
@Test
void avgThrows() {
        Calculator calc = new Calculator();
        float[] values = { };
        Exception e = assertThrows(
                        Exception.class,
                        () -> calc.avg(values),
                        "Expected div(?, 0.0) to throw, but it didn't"
                        );


        assertTrue(e.getMessage().contains("zero"));
}
```

Para rodar os testes novamente, basta executar o comando **build.sh**.

Links:

- Pergunta no StackOverflow: https://stackoverflow.com/questions/40268446/junit-5-how-to-assert-an-exception-is-thrown
- Repositório de exemplo do JUnit: https://github.com/junit-team/junit5-samples
- Repositório da disciplina: https://github.com/Luan-F/Teste_Software_2024_Leite_Luan
- Implementação dessa atividade: https://github.com/Luan-F/Teste_Software_2024_Leite_Luan/tree/main/unidade-01/atividade-01