



Teste Parametrizado e Test Suite



**Certified
Developer**
The Ultimate Tech Degree

DigitalHouse >
Coding School



Temas

1

**Teste
parametrizado**

2

Test Suite



1 | **Teste parametrizado**



Teste parametrizado

Em nossos testes, várias verificações são realizadas simplesmente para testar casos diferentes. Isso nos leva a repetir o código, como:

```
import org.junit.Assert;
import org.junit.Test;
public class MultiplicarTeste {
    @Test
    public void devemosCorroborarMultiplicacoes() {
        Assert.assertEquals(4, 2*2);
        Assert.assertEquals(6, 3*2);
        Assert.assertEquals(5, 5*1);
        Assert.assertEquals(10, 5*2);
    }
}
```



Teste parametrizado

Para construir testes parametrizados, o JUnit utiliza um *custom runner* que é o **Parameterized**.

Este nos permite definir os parâmetros de várias execuções de um único teste.

```
import org.junit.Assert;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;

import java.util.Arrays;

@RunWith(Parameterized.class)
public class MultiplicarTeste {
    @Parameterized.Parameters
    public static Iterable data(){
        return Arrays.asList(new Object[][]{
            {4,2,2},{6,3,2},{5,5,1},{10,5,2}
        });
    }
}
```



```
private int multiplierOne;
private int expected;
private int multiplierTwo;

public MultiplicarTeste(int expected, int multiplierOne, int multiplierTwo) {
    this.multiplierOne = multiplierOne;
    this.expected = expected;
    this.multiplierTwo = multiplierTwo;
}

@Test
public void deveMultiplicarOResultado(){
    Assert.assertEquals(expected,multiplierOne*multiplierTwo);
}
}
```



Teste parametrizado

Na linha de código que aparece abaixo, estamos indicando que vamos usar o runner Parameterized, que se encarregará de executar o teste quantas vezes forem necessárias dependendo da quantidade de parâmetros configurados.

```
@RunWith(Parameterized.class)
```





Teste parametrizado

A anotação **@Parameters** indica qual é o método que retornará o conjunto de parâmetros a serem usados pelo *runner*.

O que precisamos é de um construtor que permita ser inicializado com os objetos que temos em cada elemento da coleção.

Finalmente, o teste será executado usando os dados que coletamos no construtor.



2 | Test Suite



Test Suite

JUnit Test Suite nos permite agrupar e executar os testes em grupo. As *suites* de testes podem ser criadas e executadas com estas anotações:

- @RunWith
- @SuiteClasses





```
private int multiplierOne;
private int expected;
private int multiplierTwo;

public MultiplicarTeste(int expected, int multiplierOne, int multiplierTwo) {
    this.multiplierOne = multiplierOne;
    this.expected = expected;
    this.multiplierTwo = multiplierTwo;
}

@Test
public void deveMultiplicarOResultado(){
    Assert.assertEquals(expected, multiplierOne*multiplierTwo);
}
}
```



```
import junit.framework.Assert;

import org.junit.Test;

public class TestFeatureDois {
    @Test
    public void testeSegundoFeature()
    {
        Assert.assertTrue(true);
    }
}
```



Create Junit test Suite

```
import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

import com.TestFeaturePrimeiro;
import com.TestFeatureSegundo;

@RunWith(Suite.class)
@SuiteClasses({ TestFeaturePrimeiro.class, TestFeatureSegundo.class })
public class TestFeatureSuite {
    //
}
```

DigitalHouse>
Coding School