Testeo Parametrizado y Test Suite

DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

Índice

1 Testeo parametrizado

Test parametrizado

En nuestros test se realizan múltiples comprobaciones simplemente para probar diferentes casos. Esto nos lleva a repetir código, como por ejemplo:

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

Test parametrizado

Para construir test parametrizados, JUnit utiliza un *custom runner* que es **Parametrized**.

Este el cual nos permite definir los parámetros de varias ejecuciones de un solo test.

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

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

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

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

Test parametrizado

En la línea de código que aparece debajo estamos indicando que vamos a utilizar el runner de Parameterized, el cual se encargará de ejecutar el test las veces necesarias dependiendo del número de parámetros configurado.

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

Test parametrizado

La anotación **@Parameters** indica cuál es el método que nos va a devolver el conjunto de parámetros a utilizar por el *runner*.

Lo que necesitamos es un constructor que permita ser inicializado con los objetos que tenemos en cada elemento de la colección.

Finalmente, se ejecutará el test utilizando los datos que hemos recogido en el constructor.

2 | Test Suite

Test Suite

JUnit test suite nos permite agrupar y ejecutar los tests en grupo. Las *suites* de prueba se pueden crear y ejecutar con estas anotaciones:

- @RunWith
- @SuiteClasses

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

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

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

```java
import junit.framework.Assert;

import org.junit.Test;

public class TestFeatureDos {
   @Test
   public void testSegundoFeature()
   {
       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.TestFeaturePrimero;
import com.TestFeatureSegundo;

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


DigitalHouse>
Coding School