TESTS DOUBLES

Mockito

Java - Framework para tests doubles

Que es Mockito?

Mockito es un framework de test orientado a Test Double

Que es Mockito?

Mockito es un framework de test orientado a Test Double

Que es Mockito?

Mockito es un framework de test orientado a Test Double

CONCEPTOS CLAVES

01 SUT (System under test)

Cuando estamos escribiendo tes unitarios el SUT es la clase que estamos testeando.

Que es Mockito?

Mockito es un framework de test orientado a Test Double

CONCEPTOS CLAVES

01 SUT (System under test)

Cuando estamos escribiendo tes unitarios el SUT es la clase que estamos testeando.

02 DOC (Depended-on Component)

Son las dependencias conocidas por el SUT.

Que es Mockito?

Mockito es un framework de test orientado a Test Double

CONCEPTOS CLAVES

01 SUT (System under test)

Cuando estamos escribiendo tes unitarios el SUT es la clase que estamos testeando.

02 DOC (Depended-on Component)

Son las dependencias conocidas por el SUT.

¿QUE ES UN TEST DOUBLE?

Un Test double es realizar un test unitario en el cual cualquier conocimiento hacia afuera del SUT sea un objeto falso.
Existen distintos tipos de comportamiento para los Test Doubles (Stub, Mock, Spy, Dummy).

Para mas info **xUnit Test Patterns: Refactoring Test Code. Gerard Meszaros.**

Instalación

Gestor de Dependencias

Tener un gestor de dependencias en nuestro proyecto que nos permita importar las versiones del framework

Importarcion manual

Si cuenta con los archivos necesario, pueden importar de forma manual el framework

Tipos de doubles

Dummy

Son objetos que se pasan por allí, pero nunca se utilizan realmente. Por lo general, sólo son utilizadas para rellenar listas de parámetros.

Stub

Proporciona respuestas pre-programadas a las llamadas realizadas durante la prueba.

Mock

Es un Stub con la capacidad de poder verificarse.

Spy

Es un Mock pero que llama realmente a los métodos del objeto real.

Manos a la obra.
Un poco de sintaxis

Creación

ClassName varName = mock(ClassName.class);

@Mock private ClassName varName;

Set un valor

when(methodCall).thenReturn(value)

.thenReturn(value, value) .thenThrow(throwableClasses)

Verificar

verify(mock).methodCall

verify(mock, VerificationMode).methodCall

verifyNoMoreInteractions(mock)

verifyZeroInteractions(mock)

Modos de Verificacion

atLeastOne() atLeast(int)

atMost(int) times(int)

timeout(long) never()

Veamos
un ejemplo!