# devs4j

EL MEJOR SITIO WEB SOBRE PROGRAMACIÓN EN ESPAÑOL.

HOME ABOUT CONTACT

# Pruebas unitarias parte 3: Introducción a Spying con Mockito

HACE 6 DÍAS DEJA UN COMENTARIO

1 Vote

En el post anterior hablamos sobre Pruebas unitarias parte 2: JUnit y Mockito primeros pasos (http://geeks-mexico.com/2018/04/23/pruebas-unitarias-parte-2-junit-y-mockito-primeros-pasos/), el siguiente paso será hablar sobre Spying.

## ¿Qué es un Spy y cómo es diferente a un Mock?

Un Mock reemplaza por completo la clase base y solo devuelve valores por defecto o valores definidos por nosotros, mientras que un Spy mantendrá los objetos originales y solo reemplazará algunos métodos, a esto se le conoce como *partial mocking*.

Veamos el siguiente ejemplo:

```
1 @Mock
```

```
2  private ArrayList mockArrayList;
3
4  @Test
5  public void mockTest() {
6      mockArrayList.add("Object 1");
7      mockArrayList.add("Object 2");
8      mockArrayList.add("Object 3");
9      assertEquals(0, mockArrayList.size());
10 }
```

En el ejemplo anterior se crea un mock de la clase **ArrayList**, como se puede observar se ejecuta el método add 3 veces pero al realizar la validación **assertEquals**(0, mockArrayList.size()); esta devuelve verdadero y el test es exitoso.

La pregunta natural es ¿Por qué si se agregaron 3 objetos a la lista, al ejecutar el método mockArrayList.**size**() devuelve 0? La respuesta es simple, el objeto mock que creamos no contiene una implementación real de los métodos add() o size(), si no que solo su definición que será utilizada para pruebas.

En el siguiente ejemplo se mostrarán las ventajas de utilizar Spying:

```
1  @Spy
2  private ArrayList spyArrayList;
3
4  @Test
5  public void spyTest() {
6      spyArrayList.add("Object 1");
7      spyArrayList.add("Object 2");
8      spyArrayList.add("Object 3");
9
10     assertEquals(3, spyArrayList.size());
11
12     when(spyArrayList.size()).thenReturn(2
13     assertEquals(20, spyArrayList.size());
14 }
```

Como se puede ver el código es muy similar al anterior, la diferencia es que estamos utilizando un Spy en lugar de un Mock, con esto crearemos un ***partial mock***, este nos permitirá mantener el comportamiento original del objeto y solo sobre escribir algunos comportamientos, analicemos el código anterior:

1. Se agrega el objeto Object 1 a la lista
2. Se agrega el objeto Object 2 a la lista

3. Se agrega el objeto Object 3 a la lista
4. Se valida si el tamaño de la lista es 3, al estar utilizando un Spy en lugar de un Mock el resultado será verdadero así que el test sera exitoso
5. Se sobre escribe el comportamiento del método size definiendo que la próxima vez que se ejecute el valor que devolverá será 20.
6. Se valida si el tamaño de la lista es 20, como sobre escribimos el comportamiento del método size para que la próxima ejecución devuelva 20 el test será exitoso.

Como conclusión, se puede observar que es posible crear partial mocks utilizando Spying con mockito, el como usarlos en sus aplicaciones depende de ustedes.

Síguenos en nuestras redes sociales https://twitter.com/geeks_mx (https://twitter.com/geeks_mx) y https://www.facebook.com/geeksJavaMexico/ (https://www.facebook.com/geeksJavaMexico/) para recibir los siguientes posts que hablarán de temas más avanzados sobre Testing, JUnit y Mockito.

*Autor: Alejandro Agapito Bautista*

*Twitter: @raidentrance (https://geeksjavamexico.wordpress.com/mentions/raidentrance/)*

*Contacto:raidentrance@gmail.com*