Mockito – Using Spies

By Eugen Paraschiv

1. Overview

In this tutorial – we will illustrate how to make the most out of **spies in Mockito**.

We will talk about the *@Spy* annotation, how to stub a spy and, finally – we will go into the difference between *Mock* and *Spy*.

2. Simple *Spy* Example

Let's start with a simple example to see how to use a *spy*.

We use *Mockito.spy()* to **spy on a real object** – this will allow us to call all the normal methods of the object while still tracking every interaction, just as you would with a mock.

In the following example – we spy on an *ArrayList* object:

```
@Test
public void whenSpyingOnList_thenCorrect() {
    List<String> list = new ArrayList<String>
();
    List<String> spyList = Mockito.spy(list);

    spyList.add("one");
    spyList.add("two");

    Mockito.verify(spyList).add("one");
    Mockito.verify(spyList).add("two");

    assertEquals(2, spyList.size());
}
```

Note how **the real method *add()* is actually called** and how the size of *spyList* becomes 2.

3. *@Spy* Annotation

Next – let's see how to use the *@Spy* annotation. We can use *@Spy* annotation instead of *spy()* as in the following example:

```
@Spy
List<String> spyList = new ArrayList<String>();

@Test
public void whenUsingTheSpyAnnotation_thenObjectIsSpied()
{
    spyList.add("one");
    spyList.add("two");

    Mockito.verify(spyList).add("one");
    Mockito.verify(spyList).add("two");

    assertEquals(2, spyList.size());
}
```

In order to **enable Mockito annotation** (such as *@Spy*, *@Mock*, …) – we need to do one of the following:

- Call the method *MockitoAnnotations.initMocks(this)* to initialize annotated fields
- Use the built-in runner *@RunWith(MockitoJUnitRunner.class)*

4. Stubbing a *Spy*

Now – Let's see how to stub a *Spy*. We can configure/override the behavior of a method using the same syntax we would use with a mock.

In the following example – we use *doReturn()* to override the *size()* method:

```
@Test
public void whenStubASpy_thenStubbed() {
    List<String> list = new ArrayList<String>
();
    List<String> spyList = Mockito.spy(list);

    assertEquals(0, spyList.size());

    Mockito.doReturn(100).when(spyList).size();
    assertEquals(100, spyList.size());
}
```

5. *Mock* vs. *Spy in Mockito*

Finally – let's discuss the difference between *Mock* and *Spy* in Mockito – not the theoretical differences between the two concepts, just how they differ within Mockito itself.

When Mockito creates a mock – it does so from the *Class* of an Type, not from an actual instance. The mock simply creates **a bare-bones shell instance** of the Class, entirely instrumented to track interactions with it.

On the other hand, **the spy will wrap an existing instance**. It will still behave in the same way as the normal instance – the only difference is that it will also be instrumented to track all the interactions with it.

In the following example – we create a *mock* of the *ArrayList* class:

```
@Test
public void whenCreateMock_thenCreated() {
    List mockedList =
Mockito.mock(ArrayList.class);

    mockedList.add("one");
    Mockito.verify(mockedList).add("one");

    assertEquals(0, mockedList.size());
}
```

As you can see – adding an element into the mocked list doesn't actually add anything – it just calls the method with no other side-effect.

A spy on the other hand will behave differently – it will actually call the real implementation of the *add* method and add the element to the underlying list:

```
@Test
public void whenCreateSpy_thenCreate() {
    List spyList = Mockito.spy(new
ArrayList());

    spyList.add("one");
    Mockito.verify(spyList).add("one");

    assertEquals(1, spyList.size());
}
```

6. Conclusion

In this article we discussed the most useful examples of using Mockito Spies.

We learned how to create a *spy*, how to use *@Spy* annotation, how to stub a *spy* and, finally – we learned the difference between *Mock* and *Spy*.

The implementation of all these examples and code snippets **can be found in my Mockito github project** – this is an Eclipse based project, so it should be easy to import and run as it is.