Part 3: Mockito partially mocking with @Spy

Posted on September 10, 2015 by Arulkumaran Kumaraswamipillai

This extends Part 2: Mockito to fully mock the DAO layer by modifying the service layer to demonstrate partial mocking with “Mockito.spy”. In the “SimpleServiceImpl” we are interested in only testing the method “processUser(int id)” and want to mock the method “getUserById(int id)” by always returning “Sam”. So, SimpleServiceImpl is partially mocked.

Difference @Mock Vs @Spy?

@Mock is for **full mocking**, and @Spy is for **partial mocking**. When Mockito creates a mock, it does it from the class type (e.g. SimpleDao), and NOT from an actual instance. The mock simply creates a skeletal instance of the Class, which gets entirely instrumented to track interactions with it. On the other hand, the **spy** wraps an existing instance (e.g. SimpleService).

Step 1: Service and DAO layer interfaces and implementations

600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs

open all | close all

- Ice Breaker Interview
- Core Java Interview C
- JEE Interview Q&A (3
- Pressed for time? Jav
- SQL, XML, UML, JSC
- Hadoop & BigData Int
- Java Architecture Inte
- Scala Interview Q&As
- Spring, Hibernate, & M
  - Spring (18)
  - Hibernate (13)
  - AngularJS (2)
  - Git & SVN (6)
  - JMeter (2)
  - JSF (2)
  - Maven (3)
- Testing & Profiling/Sa
  - Automation Testing
  - Code Coverage (2)
  - Code Quality (2)
  - jvisualvm profiling (
  - Performance Testir

Service Layer

```
1 package com.mytutorial;
2
3 public interface SimpleService {
4     abstract String getUserById(int id);
5     abstract String processUser(int id);
6 }
7
```

```
1  package com.mytutorial;
2
3  import javax.inject.Inject;
4  import javax.inject.Named;
5
6  @Named
7  public class SimpleServiceImpl implements Simple
8
9      @Inject
10     SimpleDao dao;
11
12     public String processUser(int id) {
13         return "Hello " +  getUserById(id);
14     }
15
16     public String getUserById(int id) {
17         return dao.getNameById(id);
18     }
19 }
20
```

DAO Layer

The method “getNameById(..)” never gets invoked by the unit test as getUserById(int id) method in the service layer is mocked.

```
1 package com.mytutorial;
2
3 public interface SimpleDao {
4     abstract String getNameById(int id);
5 }
```

```
1  package com.mytutorial;
2
3  import javax.inject.Named;
4
5  @Named
6  public class SimpleDaoImpl implements SimpleDao
7
8      public String getNameById(int id) {
9          if(id == 1) {
10             return "John";
```


16 Technical Key Areas

```
11          }
12          else if (id == 2) {
13              return "Peter";
14          }
15          else {
16              throw new IllegalArgumentException()
17          }
18      }
19  }
```

Step 2: The JUnit class that partially mocks “SimpleService”, where the method “**processUser(int id)**” is tested and the method “**getUserById(int id)**” is mocked to always return “Sam”.

```
1  package com.mytutorial;
2
3  import javax.inject.Inject;
4
5  import org.junit.Assert;
6  import org.junit.Before;
7  import org.junit.Test;
8  import org.junit.runner.RunWith;
9  import org.mockito.Mockito;
10 import org.mockito.MockitoAnnotations;
11 import org.mockito.Spy;
12 import org.springframework.test.context.ContextC
13 import org.springframework.test.context.junit4.S
14
15 @ContextConfiguration(classes = { AppConfig.clas
16 @RunWith(SpringJUnit4ClassRunner.class)
17 public class SimpleTest {
18
19     @Inject
20     @Spy
21     SimpleService service;
22
23     SimpleDaoImpl dao;
24
25     @Before
26     public void setUp() {
27         MockitoAnnotations.initMocks(this);
28     }
29
30     @Test
31     public void testProcessUser() {
32         //invoke partially mocked or spied Simpl
33         Mockito.doReturn("Sam").when(service).ge
34
35         String processUser = service.processUser
36         Assert.assertEquals("Hello Sam", process
37     }
38 }
39
```

Note:

Even though id=5 is not coded for, the exception is never thrown as “SimpleService.getUserById(int id)” is never invoked, hence “SimpleDao.getNameById(int id)” never gets invoked as well.

1) **@Spy** annotation is used for partial mocking.

2) Mockito.doReturn(“Sam”).when(service).getUserById(Mockito.anyInt());

3) MockitoAnnotations.initMocks(this);

Popular Member Posts

♦ 11 Spring boot interview questions & answers

906 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

816 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

427 views

18 Java scenarios based interview Questions and Answers

409 views

♦ 7 Java debugging interview questions & answers

324 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

312 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

305 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

301 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

251 views

♦ Object equals Vs == and pass by reference Vs value

234 views

Bio | Latest Posts

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ Dozer bean mapping tutorial Dozer with Spring & Maven Tutorial ›

Posted in JUnit Mockito Spring, member-paid

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from “Blah blah” to “Wow”? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category ▼

© Disclaimer