1ockito

Mock means sample or duplicate or proxy or trail or alternate or dummy

(best)

The object that looks like real object until real obj is really arranged is called proxy/Mock object

Mockito

(It is built on the top of Junit Tool)

=>It is given to perform unit testing by mocking the Local Dependent or extenal Dependent objs..

usecase1::

Service class

|-->b.methods

(having b.logic)

---> DAO class ------> Db s/w -->Persistence methods (Persinstece logic)

class

Let us assume DAO class coding is not completed, But Service coding is completed and

to

we want finish unit testing of service class.. Then we need to create Mock obj/Fake obj/ dummy object for DAO class and assign/inject to Service class.. In order write test cases on service class methods.. (Here DAO is Local Dependent to Service class)

usecase2:

Sharekhan.com service class

BSE Comp

network

(Extenal comp Distributed App)

|

BSE: Bombay Stock Exchage

Service classs: JAva class with b.nethods having b.logic

DAO class:: JAva class with Persistence logic

(CURD operations on DB table)

|---> Data Access Object

Usecase3:

=>Service class uses the DAO class and we are ready with both service and DAO classes but we do not want to use real DAO as part of unit testing that we want to do on the Service class methods becoz the real DAO class

manipulates the real data of

DB s/w and we do not want unit testing data effecting the real DB s/w data.. For this we need to make Service class as the Isolated class by injecting proxy DAO class obj to service class as Punit testing on Service class methods..

=>When Sharkhan.com website is under development, we can not take subscription of BSE Comp becoz they charge huge money for that.. Generally this Subscription will be taken after hosting/releasing the

sharekhan.com." Till that time we need to take one mock BSE comp and assign to Service class of Sharekhan.com to perform UnitTesting.. (Here BSE comp is external dependent to Sharekhan.com)

note:: Mock objs are for temporary needs, mostly they will be used in the Unit Testing... Thse mock objs created in test methods or Test case class

but not in the real

code like

This usecase is very useful while

fixing

bugs

or while adding the updates to the project

We can do this mocking in 3 ways

service, DAO, controller classes

a) Using Mock obj/Fake obj (Provides Temporary object)

b) Using Stub obj (Providing some Functionality for the methods of mock obj like specifying for certain puts certain output should come)

c) Using Spy object (It is called Partial Mock obj/ Half mock obj that means

if u provide few functionlity to method that will be used .. otherwise

real object functionality will be used)

we

note:: While working with Spy object will be having real object also.

=> Instead of creating classes manually to prepare Mock, Stub and Spy objects.. we can use mocking frameworks available in the market whic are capable generate such classes dynamically proxy at runtime as InMemory classes (The classes that are generated in the JVM memory of RAM) [Compile time)

All these things

.java -----> .class (HDD)

(run time]

Stub, Spy objects are extension of Mock object

note:: Defining the functionality for the methods of Mock obj is called making the Mock obj as Stu object.

proxy/mock obj/stub obj

note:: Spy Object is a mock object that acts as wrapper around the Real object where the new functionality of the method will execute if defined otherwise the old functionality will execute from the real obj

This class source code and byte is permanent becoz they allobcate memory in HDD (Hard Disk Drive)

(HDD)

JVM ---> out put (JVM Memory of RAM)

Normal classes)

------>

(JVM Memory -RAM)

---Run time-------

(JVM memory -RAM)]

JVM-> output (JVM Memory)

In Memory class source code(.java) byte de

happens dynamically

at runtime in the JVM

[

Memory of the RAM

List of Mocking Frameworks::

Mockito(Popular), JMockito, EasyMock, PowerMock and etc..

Example App setup ::

======

==========

This class comes when the execution of the App begins and This class ends when the execution of the App completes (Every thing happens in JVM memory of the RAM)

All the Mocking frameworks are given on the top of Junit Framework

(note:: Mocking Frameworks are useful to get mock/proxy objs and we link them with other objs to use them in junit testcases)

[Testing LoginMgmtService class with out keeping LoginDAO class ready]

step1) create maven standalone App

File > maven Project > next --> select maven-archetype-quickstart ---> groupId : nit

ArtifactId :: MockitoUniTesting-LoginApp

default package :: com.nt.service

step2) Do following operations in pom.xml file

=>change java version to 21

=>Add the following dependencies (jars)

<dependency>

We can do Mockito programming i.e creating Mock, stub and spy objects in the test case classes of Junit in two approaches

(real bbi

proxy/mock/stub obj

a) Using Programatic Approach (The Mock objects will be created through Java code of Mockito API)

b) Using Annotation Approach (The Mock objects will be created through annotations) (Best)

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-api</artifactId>

<version>5.7.0</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.mockito/mockito-core -->

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>3.6.28</version>

<scope>test</scope>

</dependency>

impl only

step3) develop service interface, service class and DAO interface

com.nt.service

|--->|LoginMgmtService.java (1)

|--->LoginMgmtServiceImpl(c)

com.nt.dao

Enduser-> LoginService Mgmt

Step4) Develop Mickito based Test classes

step5) Run Tests...

src/main/java folder

(For complete code, refer bottom of the page)

|--->LoginDAO.java

loginDAOMock=Mockito.mock(|LoginDAO.class); // mock(-) generates InMemory class implementing

// ILoginDAO(1) having null method definitations for authenticate(-,-) nethod

@Test

public void testList() {

}

List<String> listMock-Mockito.mock(ArrayList.class); //Mock //List<String> listSpy-Mockito.spy(ArrayList.class); //Spy (or) List<String> listSpy-Mockito.spy(new ArrayList()); //Spy listMock.add("table");

Mockito.when(listMock.size()).thenReturn(10); //stub on Mock obj

listSpy.add("table");

Mockito.when(listSpy.size()).thenReturn(10); //stub on Spy obj System.out.println(listMock.size()+" "+listSpy.size());

10 0

10 if Mockito.when(...) statemetns are not commented

1

if Mockito.when(...) statemetns are

commented

Login DAO ---> DB s/w

LoginRAO

(Though the real LoginDAO class is not ready, we should complete unit testing on LoginServiceMgmt class with support Mock /Stub obj)

note:: Spy objects are useful to check how many time methods are called?.. wheather they are called or not..

```
@Test
becoz Spy object is always linked with real object.. (for this use Mockito.verify(-,-) method
public void testRegisterWithSpy() {
ILoginDAO loginDAOSpy=Mockito.spy(ILoginDAO.class);
ILoginMgmtService loginSerivce=new LoginMgmtServiceImpl(loginDAOSpy);
loginSerivce.registerUser("raja", "admin");
loginSerivce.registerUser("suresh", "visitor");
loginSerivce.registerUser("jani", "");
Mockito.verify(loginDAOSpy,Mockito.times(1)).addUser("raja", "admin");➡/ check the above register method calls addUser(-,-) is called for 1 tome Mockito.verify(loginDAOSpy,Mockito.times(0)).addUser("suresh", "visitor"); internally or not for the inputs "raja","admin"
Mockito.verify(login DAOSpy,Mockito.never()).addUser("jani", "");
}
```

Mockito Annotations

==========

@Mock --> to Genernat mock object

@Spy -->To geneate spy object

@InjectMocks-->To Inject Mock or Spy Objects Service class.

MockitoAnnotations.openMocks(this); -->call this method in @Before or constructor TestCase class in order to activate Mockito Annotations..

```
//Test case class
public class LoginMgmtServiceTestsAnno {
@InjectMocks //create the service class object
private LoginMgmtServiceImpl loginService;
@Mock // create DAO class mock object
private ILoginDAO loginDAOMock;
/*@Spy
private ILoginDAO loginDAOSpy;*/
public LoginMgmtServiceTestsAnno() {
MockitoAnnotations.openMocks(this);
}
}
```

To write stub functionalify according agile user stories /JIRA user stories (given.. when.. then..)

BDDMockito.given (login DAOMock.authenticate("raja", "rani")).willReturn(1);

equal to

```
Mockito.when(login DAOMock.authenticate("raja", "rani")).thenReturn(1);
```

Example App Code

==========

MockitoProj01-LoginApp

#src/main/java

com.nt.dao

> ILoginDAO.java

com.nt.service

> LoginMgmtService.java

>LoginMgmtServiceImpl.java

src/test/java

com.nt.tests

› Login TestsWithMockito.java

› Login TestsWithMockitoAnnotations.java

› Mock_StubVsSpyTest.java

> JRE System Library [JavaSE-17]

✓

>

>

>

Maven Dependencies

junit-jupiter-api-5.10.0.jar - C:\Users\NATARAJ\.m2\repository\org\junit\jupite

opentest4j-1.3.0.jar - C:\Users\NATARAJ\.m2\repository\org\opentest4j\opente junit-platform-commons-1.10.0.jar - C:\Users\NATARAJ\.m2\repository\org\ju

> Tapiguardian-api-1.1.2.jar - C:\Users\NATARAJ\.m2\repository\org\apiguardian

> mockito-core-5.5.0.jar - C:\Users\NATARAJ\.m2\repository\org\mockito\mock byte-buddy-1.14.6.jar - C:\Users\NATARAJ\.m2\repository\net\bytebuddy\byte byte-buddy-agent-1.14.6.jar - C:\Users\NATARAJ\.m2\repository\net\bytebud >objenesis-3.3.jar - C:\Users\NATARAJ\.m2\repository\org\objenesis\objenesis\:

>

>

> src

› target

pom.xml

//ILoginDAO.java

```
package com.nt.dao;
public interface ILoginDAO {
public int authenticate(String username, String password);
public String addUser(String username,String role);
}
```

```
//ILoginMgmtService.java
package com.nt.service;
public interface ILoginMgmtService {
}
public boolean login(String user,String pwd);
public String registerUser(String user,String role);
//service Impl class
package com.nt.service;
import com.nt.dao.ILoginDAO;
public class Login MgmtServiceImpl implements ILoginMgmtService {
private ILoginDAO dao;
public LoginMgmtServiceImpl(|LoginDAO dao) {
this.dao=dao;
}
@Override
public boolean login(String user, String pwd) {
if(user.equalsIgnoreCase("") || pwd.equalsIgnoreCase(""))
throw new IllegalArgumentException("Invalid inputs");
//use DAO
int count=dao.authenticate(user, pwd);
if (count==0)
return false;
else
return true;
}
@Override
public String registerUser(String user, String role) {
if(!user.equals("") && !role.equals("")) {
//use DAO
dao.addUser(user, role);
return "user added";
}
else
return "user not added";
}
}
package com.nt.tests;
```

```
import static org.junit.jupiter.api.Assertions.assertFalse; import static org.junit.jupiter.api.Assertions.assertThrows;
import static org.junit.jupiter.api.Assertions.assertTrue;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import org.mockito.Mockito;
import com.nt.dao.ILoginDAO;
import com.nt.service.I LoginMgmtService;
import com.nt.service.Login MgmtServiceImpl;
// Mockito in Programatic Approach
public class Login TestsWithMockito {
private static ILoginMgmtService loginService;
private static ILoginDAO login DAO;
@BeforeAll
public static void setUp() {
// Mock the LoginDAO
loginDAO=Mockito.mock(ILoginDAO.class);
System.out.println("Mock object class nanme::"+loginDAO.getClass());
//create Service class obj
loginService=new LoginMgmtServiceImpl(loginDAO);
}
@AfterAll
public static void clearDown() {
}
@Test
loginDAO=null;
loginService=null;
public void testLogin With Valid Credentials() {
// provide some funcationality for autheticate method on the mock DAO object (Stub object)
Mockito.when(loginDAO.authenticate("raja", "rani")).thenReturn(1);
// perform testing
}.
assertTrue(loginService.login("raja", "rani"));
@Test
public void testLoginWithInValidCredentials() {
// provide some funcationality for autheticate method on the mock DAO object (Stub object)
Mockito.when(loginDAO.authenticate("raja", "rani1")).thenReturn(0);
```

```
assertFalse(loginService.login("raja", "rani1"));
// perform testing
}
@Test
public void testLoginWithNoCrendentials() {
}
@Test
assertThrows(IllegalArgumentException.class, ()->loginService.login("",""));
public void testRegisterUserWithSpy() {
//create Spy Object
ILoginDAO login DAOSpy-Mockito.spy(I Login DAO.class);
// create Service class object
ILoginMgmtService loginSerivce=new LoginMgmtServiceImpl(loginDAOSpy);
}
//invoke the methods
loginSerivce.registerUser("raja", "admin");
loginSerivce.registerUser("suresh", "customer");
loginSerivce.registerUser("jani", "");
//Check weather addUser(-,-) is called expected no.of times or not
Mockito.verify(login DAOSpy, Mockito.times(1)).addUser("raja", "admin");
Mockito.verify(login DAOSpy, Mockito.times(1)).addUser("suresh", "customer");
//Mockito.verify(loginDAOSpy,Mockito.never()).addUser("jani", ""); Mockito.verify(login DAOSpy,
Mockito.times(0)).addUser("jani", "");
package com.nt.tests;
import static org.junit.jupiter.api.Assertions.assertFalse;
import static org.junit.jupiter.api.Assertions.assertThrows;
import static org.junit.jupiter.api.Assertions.assertTrue;
import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.MockitoAnnotations;
import com.nt.dao.ILogin DAO;
import com.nt.service.I LoginMgmtService;
import com.nt.service.LoginMgmtServiceImpl;
//Mockito in Programatic Approach public class Login TestsWithMockitoAnnotations { @InjectMocks //
(creates the Service class obj) private static LoginMgmtServiceImpl loginService; @Mock // creates the
MockDAO class obj
```

```
public Login TestsWithMockitoAnnotations() {
}
MockitoAnnotations.openMocks(this);
@Test
public void testLoginWithValidCredentials() {
// provide some funcationality for autheticate method on the mock DAO object (Stub object)
Mockito.when(login DAO.authenticate("raja", "rani")).thenReturn(1);
assertTrue(loginService.login("raja", "rani"));
// perform testing
}
@Test
public void testLoginWithInValidCredentials() {
// provide some funcationality for autheticate method on the mock DAO object (Stub object)
Mockito.when(login DAO.authenticate("raja", "rani1")).thenReturn(0);
assertFalse(loginService.login("raja", "rani1"));
// perform testing
}
@Test
public void testLoginWithNoCrendentials() {
}
assertThrows(IllegalArgumentException.class, ()->loginService.login("","",""));
@Test
public void testRegisterUserWithSpy() {
//create Spy Object
ILoginDAO login DAOSpy-Mockito.spy(ILoginDAO.class);
// create Service class object
ILoginMgmtService loginSerivce=new LoginMgmtServiceImpl(login DAOSpy);
//invoke the methods
loginSerivce.registerUser("raja", "admin");
loginSerivce.registerUser("suresh", "customer");
loginSerivce.registerUser("jani", "");
//Check weather addUser(-,-) is called expected no.of times or not Mockito.verify(loginDAOSpy,
Mockito.times(1)).addUser("raja", "admin");
Mockito.verify(login DAOSpy, Mockito.times(1)).addUser("suresh", "customer");
//Mockito.verify((bogin DAC SpyMotioneret)dsker(äjahi";');"); Mockito.verify(loginDAOSpy,
Mockito.times(0)).addUser("jani", "");
}
package com.nt.tests;
```

```
import java.util.ArrayList;
import java.util.List;
import org.junit.jupiter.api.Test;
import org.mockito.Mockito;
public class Mock_StubVsSpyTest {
@Test
public void testList() {
List<String> listMock-Mockito.mock(ArrayList.class); //Mock
List<String> listSpy-Mockito.spy(ArrayList.class); //Spy (or)
//List<String> listSpy-Mockito.spy(new ArrayList()); //Spy (or)
listMock.add("table");
Mockito.when(listMock.size()).thenReturn(10); //stub on Mock obj
}
listSpyattt("tradbéë);});
//test the size method
Mockito.ww/hen((l$$£$psize())) theRReturn(0));//tsuboppybobj
System.out.println(listMock.size()+" "+listSpy.size());
```