Mockito Hands-On Exercises

# Exercise 1: Mocking and Stubbing

**ExternalApi.java**

public interface ExternalApi {

String getData();

void performAction();

}

**MyService.java**

public class MyService {

private final ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

public void doAction() {

api.performAction();

}

}

**MyServiceTest.java**

import com.mycompany.mavenproject3.ExternalApi;

import com.mycompany.mavenproject3.MyService;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testExternalApi() {

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

when(mockApi.getData()).thenReturn("Mock Data");

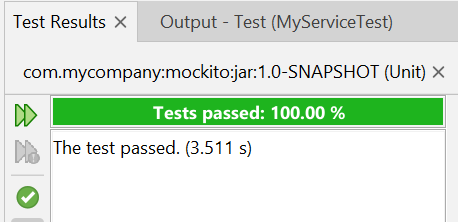
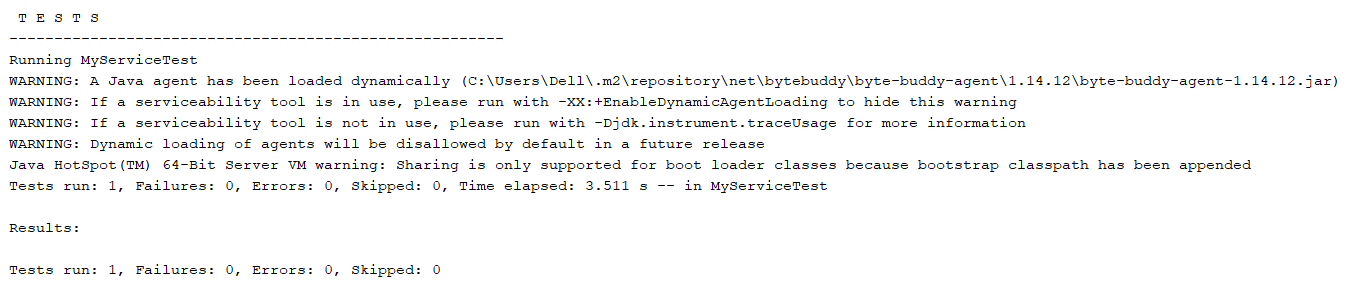
MyService service = new MyService(mockApi);

String result = service.fetchData();

assertEquals("Mock Data", result);

}

}



# Exercise 2: Verifying Interactions

**MyServiceTest.java**

import com.mycompany.mavenproject3.ExternalApi;

import com.mycompany.mavenproject3.MyService;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

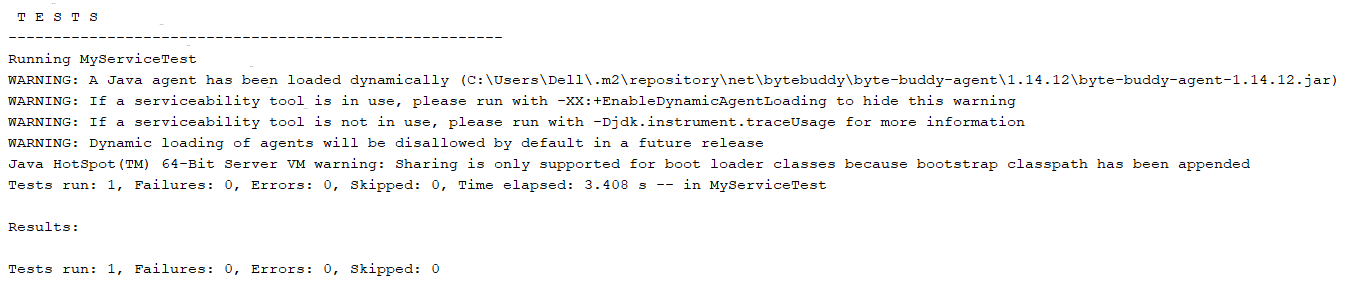
MyService service = new MyService(mockApi);

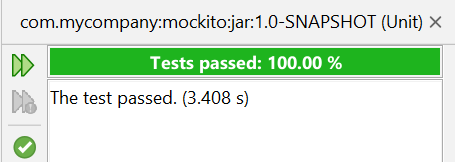
service.fetchData();

verify(mockApi).getData();

}

}





# Exercise 3: Argument Matching

**ExternalApi.java**

public interface ExternalApi {

String getData();

public void sendMessage(String eq);

}

**MyService.java**

public class MyService {

private final ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

public void sendWelcomeMessage(String name) {

api.sendMessage(name);

}

}

**MyServiceTest.java**

import com.mycompany.mavenproject3.ExternalApi;

import com.mycompany.mavenproject3.MyService;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testArgumentMatching() {

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

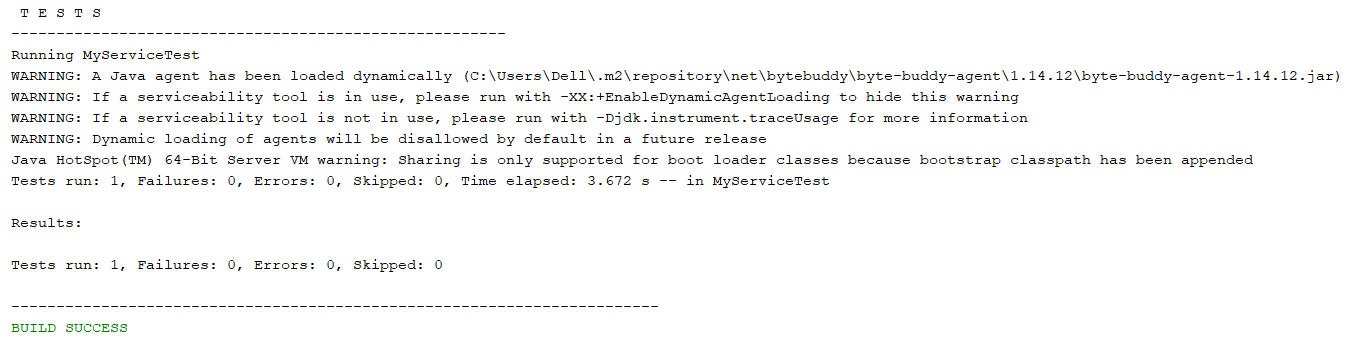
MyService service = new MyService(mockApi);

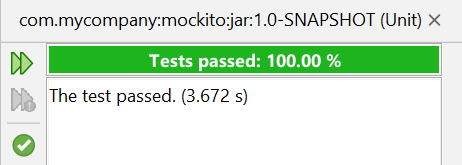
service.sendWelcomeMessage("Rupa");

verify(mockApi).sendMessage(eq("Rupa"));

}

}





# Exercise 4: Handling Void Methods

**MyServiceTest.java**

import com.mycompany.mavenproject3.ExternalApi;

import com.mycompany.mavenproject3.MyService;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testPerformActionCalled() {

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

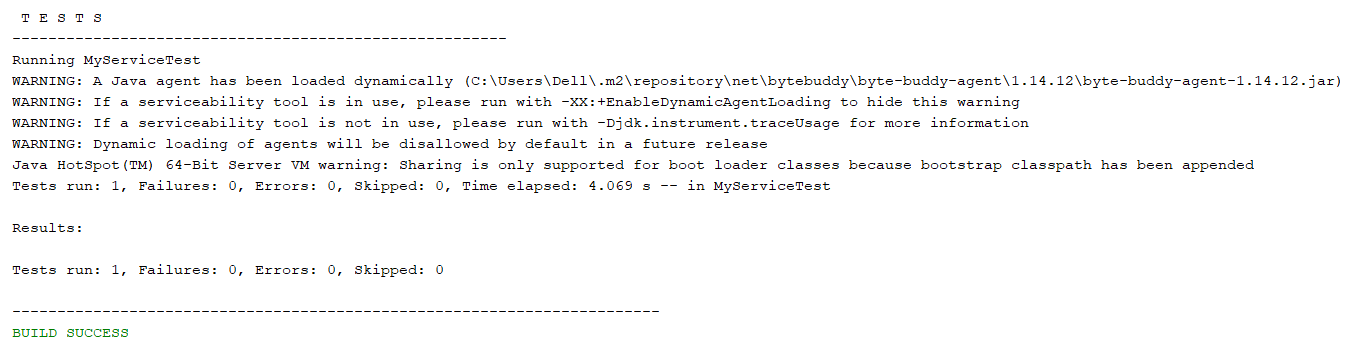
MyService service = new MyService(mockApi);

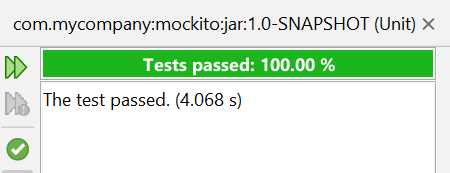
service.doAction();

verify(mockApi).performAction();

}

}





# Exercise 5: Mocking and Stubbing with Multiple Returns

**MyServiceTest.java**

import com.mycompany.mavenproject3.ExternalApi;

import com.mycompany.mavenproject3.MyService;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testMultipleReturns() {

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

when(mockApi.getData())

.thenReturn("First", "Second", "Third"); // Step 2

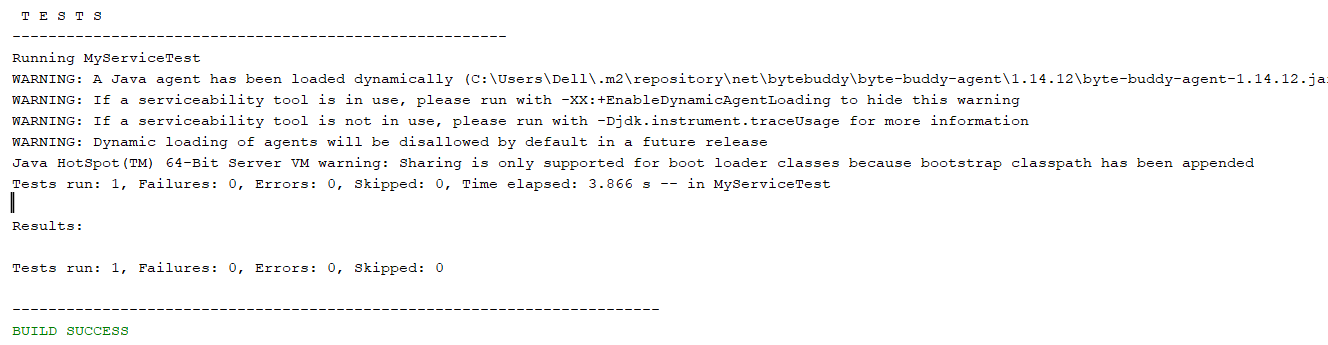
assertEquals("First", mockApi.getData());

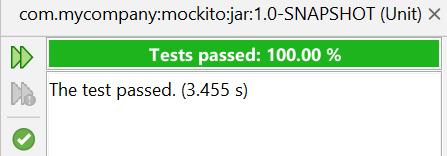
assertEquals("Second", mockApi.getData());

assertEquals("Third", mockApi.getData());

}

}





# Exercise 6: Verifying Interaction Order

**MyService.java**

public class MyService {

private final ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

public void doAction() {

api.performAction();

}

public void sendWelcomeMessage(String name) {

System.out.println("Hi, "+name);

}

public void process() {

api.getData();

api.performAction();

}

}

**MyServiceTest.java**

import com.mycompany.mavenproject3.ExternalApi;

import com.mycompany.mavenproject3.MyService;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.InOrder;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testMethodCallOrder() {

ExternalApi mockApi = mock(ExternalApi.class);

MyService service = new MyService(mockApi);

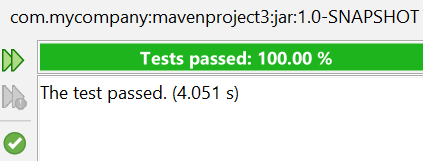
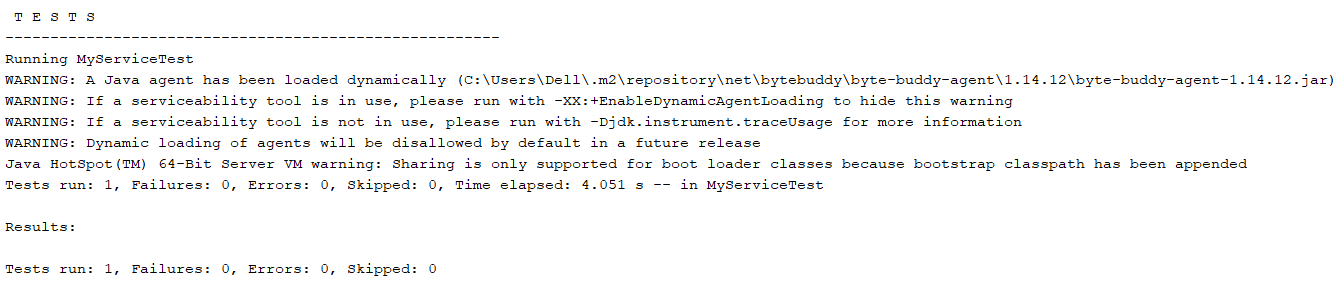
service.process(); // internally calls getData() then performAction()

InOrder inOrder = inOrder(mockApi); // Step 3

inOrder.verify(mockApi).getData(); // Step 1

inOrder.verify(mockApi).performAction(); // Step 2

}

}

# Exercise 7: Handling Void Methods with Exceptions

**MyServiceTest.java**

import com.mycompany.mavenproject3.ExternalApi;

import com.mycompany.mavenproject3.MyService;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.InOrder;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testVoidMethodThrowsException() {

ExternalApi mockApi = mock(ExternalApi.class);

// Step 2: Stub void method to throw exception

doThrow(new RuntimeException("API failed"))

.when(mockApi).performAction();

MyService service = new MyService(mockApi);

try {

service.process(); // this will call performAction()

} catch (RuntimeException e) {

// Step 3: Optional — verify that exception occurred as expected

assert(e.getMessage().equals("API failed"));

}

verify(mockApi).performAction(); // Verify it was still called

}

}