

Handson 11 : Mocking and Stubbing

Takeaways from doing this handson

1.Mocks are fake objects : They act like the real thing, but you control their behavior. It's like telling your mock.

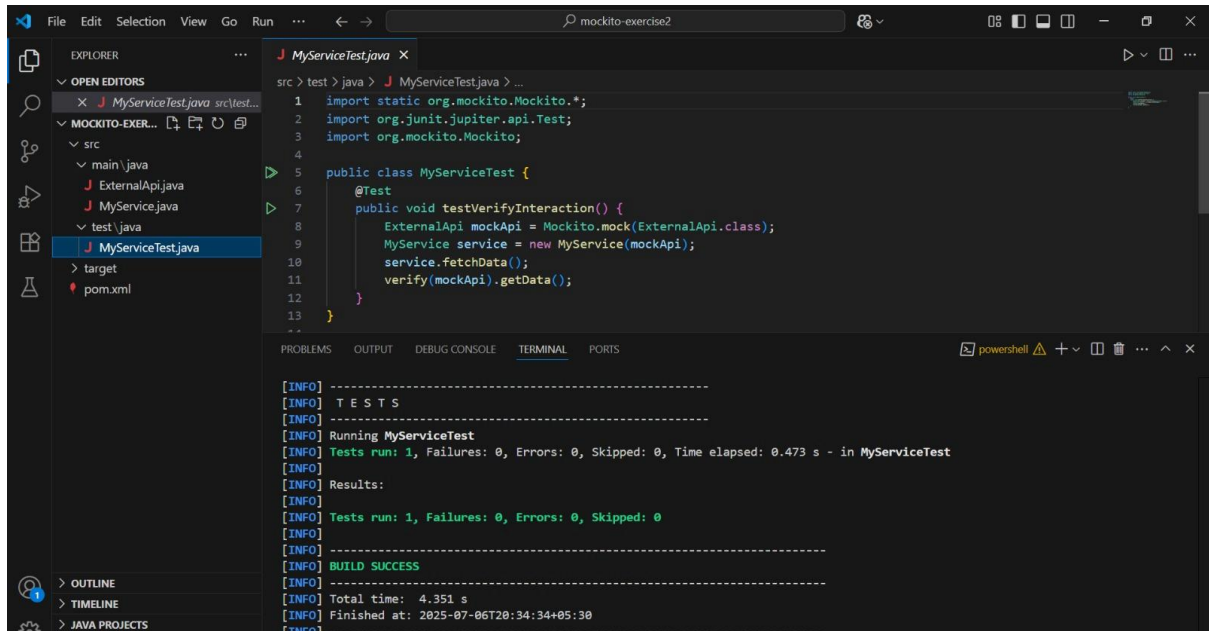
2. You can “stub” methods : That means you're pre-programming the method to return a specific value.

3. Dependency Injection : This is lowkey important “`MyService service = new MyService(mockApi)`”

injecting the mock into MyService, so the service thinks it's working with the real API — but it's actually working with the fake one you made.

4. Assertions: verifying that the service behaves correctly when the mock returns a known value.

Output snapshots:



The screenshot displays an IDE window for a project named 'mockito-exercise2'. The Explorer panel on the left shows the project structure with files like 'MyServiceTest.java' and 'pom.xml'. The main editor shows the code for 'MyServiceTest.java', which includes imports for Mockito and JUnit, and a test method 'testVerifyInteraction()'. The Terminal panel at the bottom shows the output of running the test, indicating a successful build and test execution.

```
src > test > java > J MyServiceTest.java > ...
1 import static org.mockito.Mockito.*;
2 import org.junit.jupiter.api.Test;
3 import org.mockito.Mockito;
4
5 public class MyServiceTest {
6     @Test
7     public void testVerifyInteraction() {
8         ExternalApi mockApi = Mockito.mock(ExternalApi.class);
9         MyService service = new MyService(mockApi);
10        service.fetchData();
11        verify(mockApi).getData();
12    }
13 }
```

Terminal Output:

```
[INFO] -----
[INFO] T E S T S
[INFO] -----
[INFO] Running MyServiceTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.473 s - in MyServiceTest
[INFO] Results:
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO] BUILD SUCCESS
[INFO] Total time: 4.351 s
[INFO] Finished at: 2025-07-06T20:34:34+05:30
[INFO] -----
```