Unit Testing Documentation

Owner: Ananya Ramkumar (ananyaramkumar06@gmail.com)

JUnit

JUnit is a framework that helps with writing and running your unit tests. It is a Java library that offers support for running tests and also offers extra helpers - like setup and teardown methods.

We annotate our unit tests with the **@Test** annotation in order for the tests to be recognized as such.

You would have been familiar with JUnit testing from CMSC132.

Sometime when writing JUnit tests, we may require data classes. To create those, we can use Mockito.

Mockito

Mockito is a Java-based mocking framework that is used for effective unit testing of Java applications. Mockito is used to mock interfaces so that a dummy functionality can be added to a mock interface to be used in unit testing.

The classical example for a mock object is a data provider. In production, a real database is used; during testing, a mock object simulates the database and ensures that the test conditions are always the same.

Let's explore the Mockito annotations that we use in `MvcControllerTest.java`.

What is @BeforeAll?

It is used to signal that the annotated method should be executed before all tests in the current test class. We annotate the `init()` method with @BeforeAll because we want the data within the init() to be initialized only once before running all the unit tests.

What is @BeforeEach?

It is used to signal that the annotated method should be executed before each of the unit tests. We annotate the `setup()` method with this because we want to reset our mocked objects (like `jdbcTemplate`) with `MockitoAnnotations.initMocks(this);` so that all mock interactions can be captured in the upcoming unit test.

What is @Mock?

The `@Mock` annotation allows us to create a mock object of a class or an interface. We can then use the mock to stub return values for its methods and verify if they were called. An example of stubbing is done in `setup()` where the `jdbcTemplate`'s `queryForList()` method is stubbed to always return the static `CUSTOMER1_DATA`. This stubbing gracefully handles the SQL commands made in the `updateAccountInfo()` method used frequently in `MvcController.java`.

When().Then() or When().ThenReturn()

The `thenReturn()` method lets you define the return value when a particular method of the mocked object is been called.

Mockito.verify()

Mockito Verify methods are used **to check that certain behavior happened**. We can use Mockito verify methods at the end of the testing method code to make sure that specified methods are called.

Mockito.times(x)

Mockito Times methods are used to verify that a method was called x number of times.

Further information and learning

Check out this documentation for the Mockito Framework to explore all of the different features it has: https://javadoc.io/doc/org.mockito/mockito-core/latest/org/mockito/Mockito.html

If you're a visual learner, there are tons of Mockito tutorials on YouTube:

- [https://www.youtube.com/watch?v=kXhYu939_5s&t=403s&ab_channel=JavaTechie.
- https://www.youtube.com/watch?v=--nQfs67zCM&ab_channel=ProgrammingTechie