Chapter 3 – Testing Spring Application

Different Types of testing

1. **Test Driven Development.**
2. **Unit Testing and Integration Testing**: Unit testing – testing smallest possible testable unit, while in integration testing – testing end-to-end scenario.
3. **Testing with Stubs**: There are few disadvantages of using stubs
    a. Can only be written for simpler/smaller application.
    b. Any changes made to the interface – all the implementing classes of the stub also needs to be changed. (JAVA 8 – interface default methods can come handy to overcome this limitation).
    c. Any changes made to the base level interface – all stubs needs to refactored for that change.

    Junit5 as three main modules
    1. Junit 5 platform: the junit 5 testing platform for launching Junit5 platform on the JVM
    2. Junit Jupiter Library – New library written for Junit 5
    3. Junit Vintage – Junit library providing engines that support execution of Junit 3 and Junit 4 test.

   Difference between Junit 4 and Junit 5.

4. Testing with Mocks: Mocks helps to mock the behavior of the dependent unit of code, and help testing the unit under test with ease. There are different mocking frameworks available for Java some are
    a. EasyMock: Spring was using it extensively until Spring-3 , where it embrace Mockito for its mocking needs. When Easy mock is called following things happens
        1. Declare a mock
        2. Create a mock
        3. Inject a mock
        4. Record what mock is supposed to do
        5. Tell mock actual testing that is been done
        6. Test
        7. Make sure the methods are called on the mock
        8. Validate the execution

    b. jMock : Nice and stable API for testing complex stateful logic.
        1. Declare a mock
        2. Declare and define the context of object under test , an instance of the org.jmock.Mockery class.
        3. Create a mock
        4. Inject a mock
        5. Define the expectation from the mock
        6. Test
        7. Check that mock was used

8. Validate the execution.

c. Mockito: Currently Spring prefer this mocking framework – its powerful & provides partial mock capabilities , even when real methods are invoked it can be used for verify and subbed.
   Simplest of the all –
   1. Declare and create the mock
   2. Inject mock
   3. Define the behavior of the mock
   4. Test
   5. Validate the execution

d. PowerMock : Its an extension of the easy mock, it usages a custom classloader and bytecode manipulator to test – private & static methods including constructors , final methods / Classes making it possible to test un-testable code.

Testing with Spring – SQL annotation

@Sql

@Sql annotation is used on method along with @Test annotation to specify the sql script that needs to be executed before running the test method. @Sql provides two attributes Sql.ExecutionPhase. BEFORE_TEST_METHOD [default] this specifies the script to be executed before executing the test methods. Sql.ExecutionPhase.AFTER_TEST_METHOD this specifies the script to be executed after the execution of the test method.

@Sql annotation can also be applied to class level

```
@Sql({"classpath:db/test-data-two.sql"})

@Sql(statements = {"INSERT INTO PERSON(ID, USERNAME, FIRSTNAME, LASTNAME,  PASSWORD,
HIRINGDATE, VERSION, CREATED_AT, MODIFIED_AT) VALUES (2, 'irene.adler', 'Irene', 'Adler',
'id123ds', '1990-08-18', 1, '1990-07-18', '1998-01-18');"})

@Sql(statements = "DELETE FROM PERSON", executionPhase = Sql.ExecutionPhase.AFTER_TEST_METHOD)


@Sql(scripts = {"classpath:db/person-schema.sql", "classpath:db/test-data.sql"},executionPhase
= Sql.ExecutionPhase.BEFORE_TEST_METHOD )
@SpringJUnitConfig(classes = RepositoryTest5.TestCtxConfig.class)
public class JunitTestCassName {

//….
//….
//….
}
```

@SqlConfig

@SqlConfig annotation provides information about the syntax used in SQL scripts provided as argument to an @Sql annotation.

Example

```
@Sql(scripts = "classpath:db/test-data-one.sql",
config = @SqlConfig(commentPrefix = "`", separator = "@@"))
```

@SqlGroup

@SqlGroup annotation is used for grouping multiple Sql annotations, which use different scripts/statements. You can use it on test classes or methods.

Spring – SQL testing also supports @BeforeTransation @AfterTransation, @Rollback and @Commit.

NOTE : when using @ActiveProfiles, any bean that are marked with @Profile with active profile name are loaded , and the beans that are NOT marked with @Profile are also loaded regardless if there is a active profile been mention for them or not.