Spring - Mar 15

Test

first thing first, make codes work!

security

Unit test

- part of them is working

Integration test

- put individual parts work together

JUnit test framework

Mockito framework

- mock project
- mock dependencies
- create fake objects

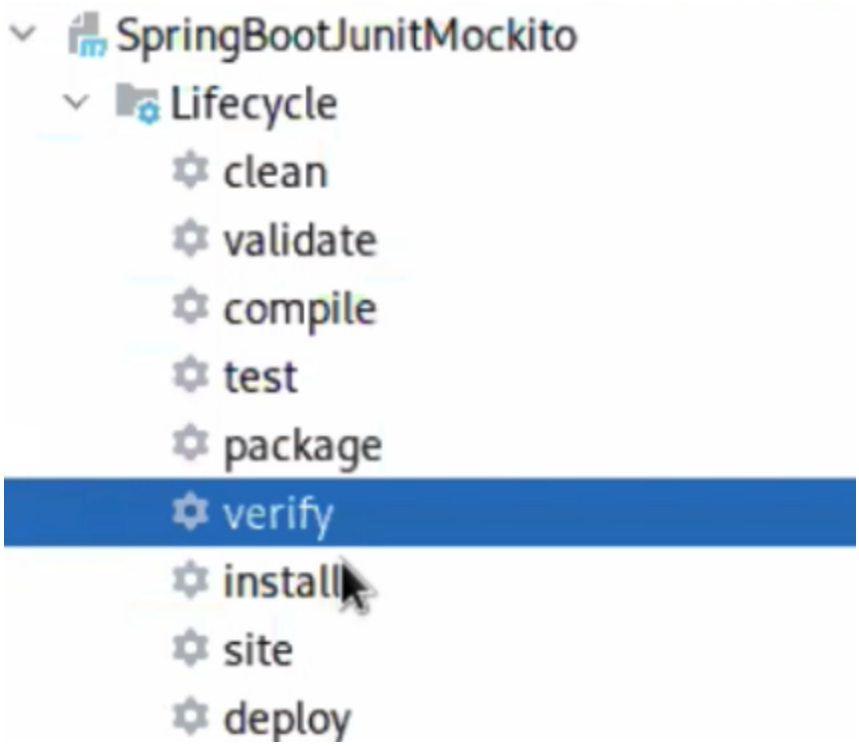
codes in the test folder only are used to build “.jar”/”.war” file

Process:

maven → run test → pass → generate “.jar”

- test functions are not run on servers

maven life circle



- whenever run maven install, it will run all previous steps and then install
- When run package, it will run all previous steps and then package

What maven do?

- when use command line run “mvn test”, it will go to test folder to find any file name with “Test” word and to execute it (or the test cases)
- “Test” should be in the front or end of class name

maven can automatically run test codes, you don’t need to run test one by one by yourself.

Recommend: **one function has only one assert method**, but it depends on team

- assertTure()
- assertFalse()
- assertEquals()

We use “import static”

```
import static org.junit.jupiter.api.Assertions.*;
```

if not use static, it will be “Assertion.assertTrue(a == b);”

```
Assertions.assertTrue(a == b);
```

life circle

```
@BeforeAll
public static void beforeClass(){
    System.out.println("Before ALL test method is executed..");
}
@BeforeEach
public void before(){
    System.out.println("Before each test method is executed..");
}
@Test
public void testMethod(){
    System.out.println("in test 1");
    assertTrue( condition: 100/2 == 50);
}
@Test
public void testMethod2(){
    System.out.println("in test 2");
    assertTrue( condition: 60/2 == 30);
}
@AfterEach
public void after(){
    System.out.println("After each test method is executed..");
```

setup method

@BeforeAll

- BeforeAll only run once when run the test class

@BeforeEach

- BeforeEach is executed every time before test method run
- clearing up environment for each test

teardown method

@AfterEach

- AfterEach is executed every time After test method run

@AfterAll

- AfterAll is only executed in the end

Attention:

- each test is independent. They don’t have relationship each other
- tests method could be ordered by use @Order annotation（default does not ordered）
  - but normally we don’t set up order, cuz if one test is failure, definitely the rest of tests are definitely failed

Demo:

SpringDemoJunitMockito

how to test in Spring without API running

- write autowired(injection bean) in the test file

```java
@SpringBootTest
public class JunitDemoTest_2InjectBean_Test {
    @Autowired
    SayHelloService sayHelloService;

    @Autowired
    SayGoodByeService sayGoodByeService;

    @Test
    public void sayHelloToNormalName() {
        assertEquals( expected: "Hello Jerry", sayHelloService.sayHelloTo( name: "Jerry"));
    }

    @Test
    public void sayHelloToEmptyName() {
        assertEquals( expected: "Hello Welcome", sayHelloService.sayHelloTo( name: ""));
    }

    @Test
    public void sayGoodByte() { assertEquals( expected: "GoodBye real", sayGoodByeService.sayGoodBye()); }

}
```

regression testing

- every time we add some new features, fix some bugs, we re-run the previous tests → to make sure everything is pass
- and we can ensure the new change can not break everything

test driven development(TDD)

- we organize our development process as we write the test **first**, **then** we change our code.

IQ: can you design some test cases? (algorithm questions, two sum)

Test case design:

design some cases don't write @Test(JUnit)(nobody asks implementation during interview)

just give them what're gonna be the input, what're gonna be the output

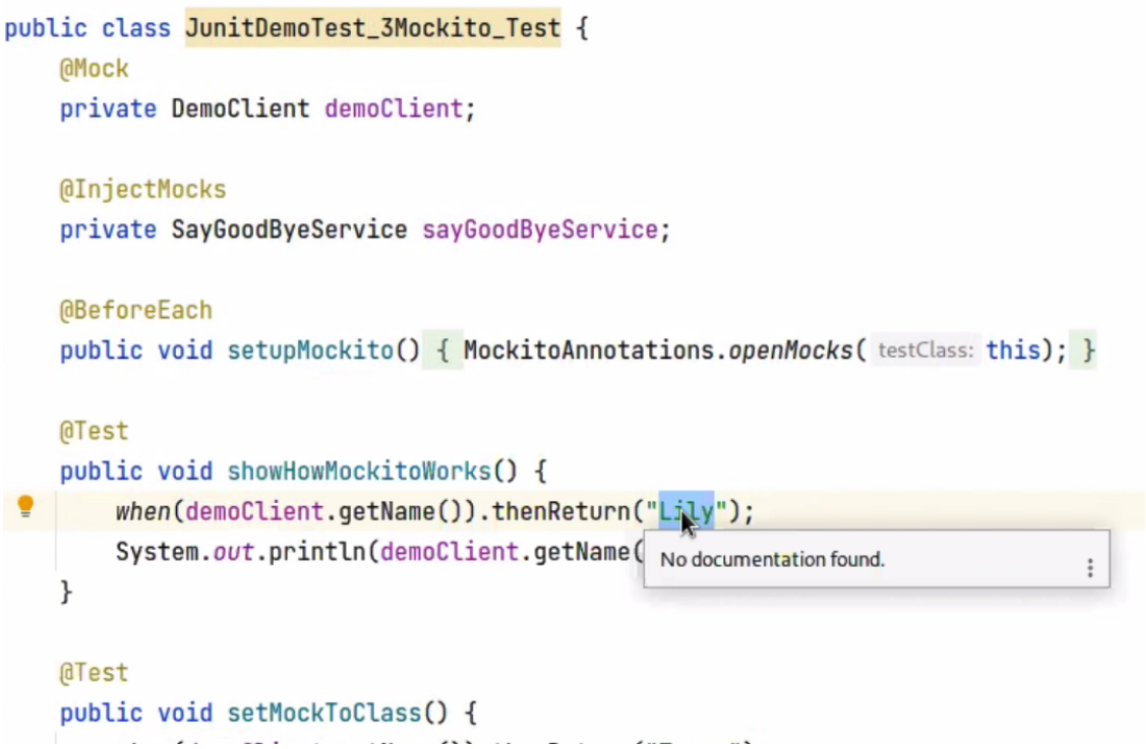
Mockito

When do unit test, there are something that are not ready or you do not want to use a real one.

- e.g want to test some errors, how to reflect to users

mockito is framework, use to simulate an object's behaviors

- create a dummy fake object that follows all the instruction we give
  - like make a DAO to throw exception
- use
  -



  - @Mock annotation
  - @InjectMocks annotation

        - to inject the mock demoObject
    - give an instruction

```
@BeforeEach
public void setupMockito() { MockitoAnnotations.openMocks( testClass: this); }
```

```
@Test
public void showHowMockitoWorks() {
    when(demoClient.getName()).thenReturn("Lily");
    System.out.println(demoClient.getName());
}
```

When demoClient.getName() is called, then return “Lily”

- thenReturn() return a value
- thenThrow() return/throw an exception
    - to test service whether can handle the exception

IQ: What is Mockito? What’s the purpose? Why use Mockito?

Why do we need mock? Why do we just directly test codes?

- because in the real world situation is you have a lot of dependencies. dependencies are easy to satisfy. Some of the dependencies are very hard to control like, like the database.
- like database shut down, → we use develop environment database → remote connection

@Spy → hybrid version of a real object and fake object

```java
@Spy
private SayHelloService sayHelloService;

@BeforeEach
public void setupMockito() {
    MockitoAnnotations.openMocks( testClass: this);
}

@Test
public void testHi() {
    System.out.println(this.sayHelloService.sayHi( name: "Dawei"));
    doReturn( toBeReturned: "Hi Daniel").when(sayHelloService).sayHi(anyString());
    System.out.println(this.sayHelloService.sayHi( name: "Dawei"));
}
```

- When the first line of testHi(), it uses the real method of the object
- the second line, we give an instruction to it

IQ: ask any testing experience

- JUnit Mockito
- how to trigger it test process by maven

API test

Integration test

directly use real servers, real databases

Unit test

- because the test is mock, we call it unit test

- MicroServices, one service could be one unit
- In that case, we're going to use a real deal with real database. And that could also be a unit test. It's the scope like what is the real unit has what is the integration test? So it is because the forecasts are different? It's not because of the unit. The value is you're testing for class or you're testing or for service that

caused the unit. But, you know, just, again, go to a team, whatever the call unit has to, you know, what's the scope of air testing scope? A lot of these tests are so the hybrid. Some of the dependencies are marked some of its dependencies are real.

sonarqube

Code Quality and Code Security Tool

Github - public

- review codes
- create (new)pull request

Gitlab - private