Learning Consolidation

Test RESTful Services at Service Layer and Data Layer by Using Testing Tools (JUnit, Mockito)

In this Sprint, you have learned to:

- Describe JUnit 5 Testing
- Implement repository layer testing
- Explore the Mockito framework

Slide Note

JUnit 5 Architecture

- `JUnit Platform`
  - Launches testing frameworks on the JVM
  - Its TestEngine API is used to build a testing framework that runs on the JUnit platform
- `JUnit Jupiter`
  - Blend of a new programming model for writing tests and an extension model for extensions
  - Contains new annotations like `@BeforeEach`, `@AfterEach`, `@AfterAll`, `@BeforeAll`, etc.
- `JUnit Vintage`
  - Provides support to execute previous JUnit version 3 and 4 tests on this new platform

Slide Note

JUnit 5 Annotations

- In JUnit 5, the test lifecycle is driven by 4 primary annotations i.e., @BeforeEach,@AfterEach, @BeforeAll, and @AfterAll. Along with it, each test method must be marked with the @Test annotation.

JUnit 5	Description
@Test	Identifies a method as a test method.
@BeforeEach	Denotes that the annotated method should be executed *before* **each** @Test, @RepeatedTest method, etc., in the current class; analogous to JUnit 4's @Before.
@AfterEach	Denotes that the annotated method should be executed *after* **each** @Test, @RepeatedTest method, etc. in the current class; analogous to JUnit 4's @After.

Slide Note

JUnit 5 Annotations (contd.)

JUnit	Description
@Ignore or @Ignore("Why disabled")	This is useful when the underlying code has been changed and the test case has not yet been adopted. If the execution time of this test is too long to be included, then it is best to provide an optional description of why the test is disabled.
@Test (timeout=100)	Fails if the method takes longer than 100 milliseconds.

Slide Note

```
@Test
@DisplayName("Test case for saving customer object")
void givenCustomerToSaveShouldReturnSavedCustomer() {
    customerRepository.save(customer);
    Customer customer1 = customerRepository.findById(customer.getCustomerId()).get();
    assertNotNull(customer1);
    assertEquals(customer.getCustomerId(), customer1.getCustomerId());
}
```

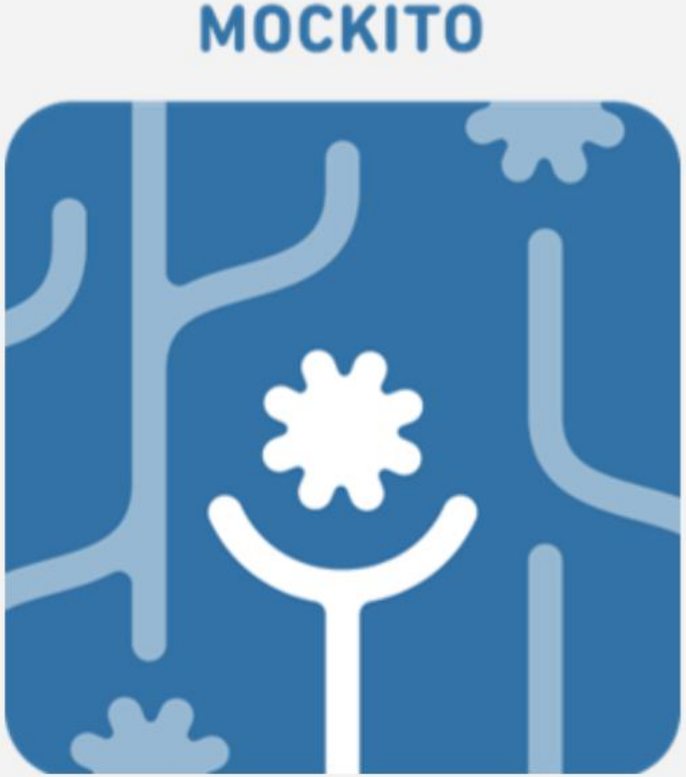
Testing Repository Layer

- In the application, the repository layer is dependent on the external database MongoDB.
- Call the `save()` method of `CustomerRepository` and save the customer object in the database.
- Here, Spring object is getting saved in an external database.
- For this reason, we call the repository layer testing an "integration testing."
- Integration tests verify that the code works with external dependencies correctly.
- Integration testing may detect errors when modules are integrated to build the overall system.

Slide Note

Mockito Framework

- Mockito is a popular mock framework that can be used in conjunction with the JUnit.
- Mockito allows us to create and configure mock objects. Using Mockito significantly simplifies the development for classes with external dependencies.
- When we use Mockito in tests, we can:
  - Mock away external dependencies and insert the mocks into the code under test.
  - Execute the code under test.
  - Validate that the code is executed correctly.



Slide Note

Testing Service Layer With Mockito (contd.)

The `when …. Then` is the crux of Mockito testing.

```
@Test
public void givenCustomerToSaveReturnSavedCustomerSuccess() throws CustomerAlreadyExistsException {
    when(customerRepository.findById(customer1.getCustomerId()) ).thenReturn(Optional.ofNullable( value: null));
    when(customerRepository.save(any())).thenReturn(customer1);
    assertEquals(customer1,customerService.saveCustomerDetail(customer1));
    verify(customerRepository,times( wantedNumberOfInvocations: 1)).save(any());
    verify(customerRepository,times( wantedNumberOfInvocations: 1)).findById(any());
}
```

When a new object is to be saved in the database, the assumption is that the object is not present, so when the `findById` method is used, the expectation is that null must be returned. This is done by the `when` method of the Mockito. It mocks the repository layer and sets an expectation.

- When `findById` is called the expectation is to return a null object.
- Similarly, while saving the object, the expectation is that the object will get saved and the saved object will be return.
- The when method of Mockito is not calling the customeRepository in real it's just going to `mock` it.
- The verify method check how many times mock method is getting called.