Testing

Service Layer Unit Testing

- To test any class , first open the test package and in the test class annotate it with `@SpringBootTest`
- Create an instance of the class which you want to test and inject the dependency by using `@Autowired`
- If the class which u need to test needs the functions of other classes to work , then , create an instance of that class which is needed and annotate it with `@MockBean`
- void setUp() : this method is used to initialize or preapre any resource or configurations needed before running any tests. This is where you define the contents of the mockbean. It has the following annotations :
    - `@BeforeEach` : Runs the annotated method **before each test method**.
    - `@BeforeAll` : Runs the annotated method **once before all tests in the class**.
    - `@AfterEach` : Runs the annotated method **after each test method**.
    - `@AfterAll` : Runs the annotated method **once after all tests in the class**.

Example code :

```
@SpringBootTest
class DepartmentServiceTest {

    @Autowired
    private DepartmentService departmentService;

    @MockBean
    private DepartmentRepository departmentRepository;
    @BeforeEach
    void setUp() {
        Department department = Department.builder()
        .departmentName("IT")
        .departmentAddress("Here")
        .departmentCode("065")
        .departmentId(1L)
        .build();

        Mockito.when(departmentRepository.findByDepartmentName( "IT")).thenRe
    }



    @Test
    @DisplayName("Get Data based on valid deaprtment  name")
    public void whenValidDepartmentName_thenDepartmentShouldFound(){
        String departmentname = "IT";
        Department  found = departmentService.fetchDepartmentByName(departm
        assertEquals(departmentname,found.getDepartmentName());
    }
}
```

What Happens:

- The test calls `departmentService.fetchDepartmentByName("IT")`.
- This method interacts with the mocked `departmentRepository` to fetch the `Department` object.
- The `assertEquals` statement verifies that the department name matches the expected result.

`fetchDepartmentByName` Method (Service Layer)

- **What Happens**:
    - Calls the repository's `findByDepartmentName(departmentName)` method to fetch the department.
    - Returns the department object.

Order of Execution

1. **Test Initialization**:
    - Spring Boot initializes the test environment.
    - The mock `DepartmentRepository` replaces the real bean.
2. **Setup (`@BeforeEach`)**:
    - A mock response for `findByDepartmentName("IT")` is configured using `Mockito.when`.
3. **Test Execution**:
    - The test calls `fetchDepartmentByName("IT")` on the service.
    - The service calls the mock repository.
    - The mock repository returns the pre-configured department object.
    - The `assertEquals` validates the result.

```
Mockito.when(departmentRepository.findByDepartmentName( "IT")).thenReturn
```

`Mockito.when` : When `findByDepartmentName("IT")` is called, it returns the pre-configured `department`

Builder Pattern

- The `Department.builder()` is a Lombok feature (`@Builder`) that provides a fluent way to create an object.

`@DisplayName` : Provides a custom display name for the test case

`@Disabled` : to disable a single test case whenever required

Repo Layer Testing

```
@DataJpaTest
class DepartmentRepositoryTest {
    @Autowired
    private DepartmentRepository departmentRepository;

    @Autowired
    private TestEntityManager entityManager;
    @BeforeEach
    void setUp() {
        Department department = Department.builder()
                .departmentName("Mech")
                .departmentAddress("Delhi")
                .departmentCode("OO1")
                .build();
        entityManager.persist(department);
    }

    @Test
    public void whenFindbyId_thenReturnDepartment(){
        Department department = departmentRepository.findById(1L).get();
        assertEquals(department.getDepartmentName(),"Mech");
    }
}
```

- **`@DataJpaTest`**: Configures a test environment optimized for testing JPA repositories (e.g., in-memory database).
- **`TestEntityManager`**: A helper provided by Spring for directly interacting with the database during tests.
- **`@BeforeEach`** - **`setUp()`**: Sets up test data by persisting a `Department` entity into the in-memory database before each test.
- **`@Test - whenFindById_thenReturnDepartment()`**: Tests the repository's `findById()` method to verify it retrieves the correct department entity.

- **Assertions (`assertEquals`)**: Checks if the retrieved department's name matches the expected value ("Mech").

Controller Layer Testing

```
@WebMvcTest(DepartmentController.class)
class DepartmentControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @MockBean
    private DepartmentService departmentService;

    private Department department;

    @BeforeEach
    void setUp() {
        Department department = Department.builder()
                .departmentName("IT")
                .departmentAddress("IDK")
                .departmentCode("oo1")
                .departmentId(1L)
                .build();
    }
    @Test
    void saveDepartment() throws Exception {
        Department inputDepartment = Department.builder()
                .departmentName("IT")
                .departmentAddress("IDK")
                .departmentCode("oo1")
                .build();
        Mockito.when(departmentService.saveDepartment(inputDepartment)).then
        mockMvc.perform(MockMvcRequestBuilders.post("/departments"))
                .contentType(MediaType.APPLICATION_JSON)
                .content("{\n" +
                        "\t\"departmentName\":\"IT\",\n" +
                        "\t\"departmentAddress\":\"IDK\",\n" +
                        "\t\"departmentCode\":\"oo1\"\n" +
                        "}")
                .andExpect(MockMvcResultMatchers.status().isOk());
    }

    @Test
    void fDepartmentById() throws Exception {
        Mockito.when(departmentService.fetchDepartmentById(1L)).thenReturn(de
        mockMvc.perform(MockMvcRequestBuilders.get("/departments/1"))
                .contentType(MediaType.APPLICATION_JSON)
                .andExpect(status().isOk());
    }
}
```

JWT

Code available understand and chatgpt later - Generating and Validating

Google and Github Login

Left over topics :

Email Verification

Forget Password

Change password