Superset ID: 6416829

Week 2

# Spring Testing Exercises

**Exercise 1: Basic Unit Test for a Service Method**

@Service  
public class CalculatorService {  
 public int add(int a, int b) {  
 return a + b;  
 }  
}  
@Test  
public void testAdd() {  
 assertEquals(5, calculatorService.add(2, 3));  
}  
 **Output:**  
Test passed: testAdd

**Exercise 2: Mocking a Repository in a Service Test**

@Mock  
private UserRepository userRepository;  
  
@InjectMocks  
private UserService userService;  
  
@Test  
public void testGetUserById() {  
 User user = new User();  
 user.setId(1L);  
 user.setName("Alice");  
  
 when(userRepository.findById(1L)).thenReturn(Optional.of(user));  
 User result = userService.getUserById(1L);  
 assertEquals("Alice", result.getName());  
}  
 **Output:**  
Test passed: testGetUserById

**Exercise 3: Testing a REST Controller with MockMvc**

@WebMvcTest(UserController.class)  
public class UserControllerTest {  
  
 @Autowired  
 private MockMvc mockMvc;  
  
 @MockBean  
 private UserService userService;  
  
 @Test  
 public void testGetUser() throws Exception {  
 User user = new User();  
 user.setId(1L);  
 user.setName("Alice");  
  
 when(userService.getUserById(1L)).thenReturn(user);  
 mockMvc.perform(get("/users/1"))  
 .andExpect(status().isOk())  
 .andExpect(jsonPath("$.name").value("Alice"));  
 }  
}  
 **Output:**  
MockMvc test passed for /users/1

**Exercise 4: Integration Test with Spring Boot**

@SpringBootTest  
@AutoConfigureMockMvc  
public class UserIntegrationTest {  
  
 @Autowired  
 private MockMvc mockMvc;  
  
 @Autowired  
 private UserRepository userRepository;  
  
 @BeforeEach  
 public void setup() {  
 userRepository.deleteAll();  
 User user = new User();  
 user.setId(1L);  
 user.setName("Alice");  
 userRepository.save(user);  
 }  
  
 @Test  
 public void testGetUserIntegration() throws Exception {  
 mockMvc.perform(get("/users/1"))  
 .andExpect(status().isOk())  
 .andExpect(jsonPath("$.name").value("Alice"));  
 }  
}  
 **Output:**  
Integration test passed: /users/1 returns Alice

**Exercise 5: Test Controller POST Endpoint**

@WebMvcTest(UserController.class)  
public class CreateUserControllerTest {  
  
 @Autowired  
 private MockMvc mockMvc;  
  
 @MockBean  
 private UserService userService;  
  
 @Test  
 public void testCreateUser() throws Exception {  
 User user = new User(1L, "Bob");  
 when(userService.saveUser(any(User.class))).thenReturn(user);  
  
 mockMvc.perform(post("/users")  
 .contentType(MediaType.APPLICATION\_JSON)  
 .content("{\"id\":1,\"name\":\"Bob\"}"))  
 .andExpect(status().isOk())  
 .andExpect(jsonPath("$.name").value("Bob"));  
 }  
}  
 **Output:**  
POST /users test passed: returns Bob

**Exercise 6: Test Service Exception Handling**

@ExtendWith(MockitoExtension.class)  
public class UserServiceExceptionTest {  
  
 @Mock  
 private UserRepository userRepository;  
  
 @InjectMocks  
 private UserService userService;  
  
 @Test  
 public void testUserNotFound() {  
 when(userRepository.findById(100L)).thenReturn(Optional.empty());  
 User result = userService.getUserById(100L);  
 assertNull(result);  
 }  
}  
 **Output:**  
Test passed: user not found returns null

**Exercise 7: Test Custom Repository Query**

@DataJpaTest  
public class UserRepositoryTest {  
  
 @Autowired  
 private UserRepository userRepository;  
  
 @BeforeEach  
 public void setup() {  
 userRepository.save(new User(1L, "John"));  
 userRepository.save(new User(2L, "John"));  
 userRepository.save(new User(3L, "Alice"));  
 }  
  
 @Test  
 public void testFindByName() {  
 List<User> users = userRepository.findByName("John");  
 assertEquals(2, users.size());  
 }  
}  
 **Output:**  
Test passed: 2 users found with name John

**Exercise 8: Test Controller Exception Handling**

@WebMvcTest(UserController.class)  
public class ExceptionHandlerTest {  
  
 @Autowired  
 private MockMvc mockMvc;  
  
 @MockBean  
 private UserService userService;  
  
 @Test  
 public void testUserNotFoundException() throws Exception {  
 when(userService.getUserById(1L)).thenThrow(new NoSuchElementException());  
  
 mockMvc.perform(get("/users/1"))  
 .andExpect(status().isNotFound())  
 .andExpect(content().string("User not found"));  
 }  
}  
**Output:**  
Exception test passed: NoSuchElementException handled with 404

**Exercise 9: Parameterized Test with JUnit**

class CalculatorServiceParamTest {  
  
 private final CalculatorService service = new CalculatorService();  
  
 @ParameterizedTest  
 @CsvSource({  
 "1,2,3",  
 "2,3,5",  
 "-1,-1,-2",  
 "0,0,0"  
 })  
 void testAdd(int a, int b, int expected) {  
 assertEquals(expected, service.add(a, b));  
 }  
}  
**Output:**  
Test passed: All add() combinations