Mocking Dependencies in Spring Tests using Mockito HandsOn

Exercise 1: Mocking a Service Dependency in a Controller Test

Task: Write a unit test for a Spring controller that uses a service to fetch data. Mock the service dependency using Mockito.

Code :

@Entity

**public** **class** User {

@Id

**private** Long id;

**private** String name;

// getters and setters

}

@Service

**public** **class** UserService {

@Autowired

**private** UserRepository userRepository;

**public** User getUserById(Long id) {

**return** userRepository.findById(id).orElse(**null**);

}

}

@RestController

@RequestMapping("/users")

**public** **class** UserController {

@Autowired

**private** UserService userService;

@GetMapping("/{id}")

**public** ResponseEntity<User> getUser(@PathVariable Long id) {

**return** ResponseEntity.ok(userService.getUserById(id));

}

}

UserControllerTest.java

**import** **static** org.mockito.ArgumentMatchers.*anyLong*;

**import** **static** org.mockito.Mockito.*when*;

**import** **static** org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultHandlers.print;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

@WebMvcTest(UserController.**class**)

**class** UserControllerTest {

@Autowired

**private** MockMvc mockMvc;

@MockBean

**private** UserService userService;

@Autowired

**private** ObjectMapper objectMapper;

@Test

**void** testGetUser\_Success() **throws** Exception {

// Given

Long userId = 1L;

User mockUser = **new** User();

mockUser.setId(userId);

mockUser.setName("John Doe");

// Mock the service behavior

*when*(userService.getUserById(userId)).thenReturn(mockUser);

// When & Then

mockMvc.perform(get("/users/{id}", userId)

.contentType(MediaType.APPLICATION\_JSON))

.andDo(print())

.andExpect(status().isOk())

.andExpect(content().contentType(MediaType.APPLICATION\_JSON))

.andExpect(jsonPath("$.id").value(userId))

.andExpect(jsonPath("$.name").value("John Doe"));

// Verify service method was called

Mockito.*verify*(userService, Mockito.*times*(1)).getUserById(userId);

}

}

Exercise 2: Mocking a Repository in a Service Test

Task: Write a unit test for a Spring service that uses a repository to fetch data. Mock the repository dependency using Mockito.

Code :

@Entity

**public** **class** User {

@Id

**private** Long id;

**private** String name;

// getters and setters

}

**public** **interface** UserRepository **extends** JpaRepository<User, Long> {

}

@Service

**public** **class** UserService {

@Autowired

**private** UserRepository userRepository;

**public** User getUserById(Long id) {

**return** userRepository.findById(id).orElse(**null**);

}

}

UserServiceTest.java

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

**import** **static** org.mockito.ArgumentMatchers.*anyLong*;

**import** **static** org.mockito.Mockito.\*;

@ExtendWith(MockitoExtension.**class**)

**class** UserServiceTest {

@Mock

**private** UserRepository userRepository;

@InjectMocks

**private** UserService userService;

**private** User testUser;

@BeforeEach

**void** setUp() {

testUser = **new** User();

testUser.setId(1L);

testUser.setName("John Doe");

}

@Test

**void** testGetUserById\_Success() {

// Given

Long userId = 1L;

*when*(userRepository.findById(userId)).thenReturn(Optional.*of*(testUser));

// When

User result = userService.getUserById(userId);

// Then

*assertNotNull*(result);

*assertEquals*(userId, result.getId());

*assertEquals*("John Doe", result.getName());

// Verify repository method was called exactly once

*verify*(userRepository, *times*(1)).findById(userId);

}

}

Exercise 3: Mocking a Service Dependency in an Integration Test

Task: Write an integration test for a Spring Boot application that mocks a service dependency using Mockito.

Code :

@Entity

**public** **class** User {

@Id

**private** Long id;

**private** String name;

// getters and setters

}

@Service

**public** **class** UserService {

@Autowired

**private** UserRepository userRepository;

**public** User getUserById(Long id) {

**return** userRepository.findById(id).orElse(**null**);

}

}

@RestController

@RequestMapping("/users")

**public** **class** UserController {

@Autowired

**private** UserService userService;

@GetMapping("/{id}")

**public** ResponseEntity<User> getUser(@PathVariable Long id) {

**return** ResponseEntity.ok(userService.getUserById(id));

}

}

UserIntegrationTest.java

**import** **static** org.mockito.ArgumentMatchers.*anyLong*;

**import** **static** org.mockito.Mockito.\*;

**import** **static** org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultHandlers.print;

**import** **static** org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

@SpringBootTest

@AutoConfigureMockMvc

**class** UserIntegrationTest {

@Autowired

**private** MockMvc mockMvc;

@MockBean

**private** UserService userService;

@Autowired

**private** ObjectMapper objectMapper;

**private** User testUser;

@BeforeEach

**void** setUp() {

testUser = **new** User();

testUser.setId(1L);

testUser.setName("John Doe");

// Reset all mocks before each test

Mockito.reset(userService);

}

@Test

**void** testGetUser\_Success\_FullIntegration() **throws** Exception {

// Given

Long userId = 1L;

*when*(userService.getUserById(userId)).thenReturn(testUser);

// When & Then

ResultActions result = mockMvc.perform(get("/users/{id}", userId)

.contentType(MediaType.APPLICATION\_JSON)

.accept(MediaType.APPLICATION\_JSON))

.andDo(print())

.andExpect(status().isOk())

.andExpect(content().contentType(MediaType.APPLICATION\_JSON))

.andExpect(jsonPath("$.id").value(userId))

.andExpect(jsonPath("$.name").value("John Doe"));

// Verify service interaction

*verify*(userService, *times*(1)).getUserById(userId);

*verifyNoMoreInteractions*(userService);

}

}