Mockito - Mock Dependencies Exercises: Solutions

# 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.

## Step-by-Step Solution:

1. Create the User Entity:

@Entity  
public class User {  
 @Id  
 private Long id;  
 private String name;  
 // getters and setters  
}

2. Create the UserService:

@Service  
public class UserService {  
 @Autowired  
 private UserRepository userRepository;  
  
 public User getUserById(Long id) {  
 return userRepository.findById(id).orElse(null);  
 }  
}

3. Create the UserController:

@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));  
 }  
}

4. Create the UserControllerTest:

@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("Test User");  
  
 Mockito.when(userService.getUserById(1L)).thenReturn(user);  
  
 mockMvc.perform(get("/users/1"))  
 .andExpect(status().isOk())  
 .andExpect(jsonPath("$.name").value("Test User"));  
 }  
}

# 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.

## Step-by-Step Solution:

1. Create the User Entity (same as above).

2. Create the UserRepository:

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

3. Create the UserService (same as above).

4. Create the UserServiceTest:

@ExtendWith(MockitoExtension.class)  
public class UserServiceTest {  
  
 @Mock  
 private UserRepository userRepository;  
  
 @InjectMocks  
 private UserService userService;  
  
 @Test  
 public void testGetUserById() {  
 User user = new User();  
 user.setId(1L);  
 user.setName("Test User");  
  
 Mockito.when(userRepository.findById(1L)).thenReturn(Optional.of(user));  
  
 User result = userService.getUserById(1L);  
 assertEquals("Test User", result.getName());  
 }  
}

# 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.

## Step-by-Step Solution:

1. Create the User Entity (same as above).

2. Create the UserService (same as above).

3. Create the UserController (same as above).

4. Create the UserIntegrationTest:

@SpringBootTest  
@AutoConfigureMockMvc  
public class UserIntegrationTest {  
  
 @Autowired  
 private MockMvc mockMvc;  
  
 @MockBean  
 private UserService userService;  
  
 @Test  
 public void testGetUser() throws Exception {  
 User user = new User();  
 user.setId(1L);  
 user.setName("Integration User");  
  
 Mockito.when(userService.getUserById(1L)).thenReturn(user);  
  
 mockMvc.perform(get("/users/1"))  
 .andExpect(status().isOk())  
 .andExpect(jsonPath("$.name").value("Integration User"));  
 }  
}