# Mockito Mock Dependencies Exercises - Solutions

## Exercise 1: Mocking a Service Dependency in a Controller Test

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;  
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;  
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.jsonPath;  
import org.junit.jupiter.api.Test;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;  
import org.springframework.boot.test.mock.mockito.MockBean;  
import org.springframework.test.web.servlet.MockMvc;  
import static org.mockito.Mockito.\*;  
  
@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("John Doe");  
  
 when(userService.getUserById(1L)).thenReturn(user);  
  
 mockMvc.perform(get("/users/1"))  
 .andExpect(status().isOk())  
 .andExpect(jsonPath("$.name").value("John Doe"));  
 }  
}

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

import static org.junit.jupiter.api.Assertions.\*;  
import static org.mockito.Mockito.\*;  
import java.util.Optional;  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
import org.mockito.InjectMocks;  
import org.mockito.Mock;  
import org.mockito.MockitoAnnotations;  
  
public class UserServiceTest {  
  
 @Mock  
 private UserRepository userRepository;  
  
 @InjectMocks  
 private UserService userService;  
  
 @BeforeEach  
 public void setUp() {  
 MockitoAnnotations.openMocks(this);  
 }  
  
 @Test  
 public void testGetUserById() {  
 User user = new User();  
 user.setId(1L);  
 user.setName("John Doe");  
  
 when(userRepository.findById(1L)).thenReturn(Optional.of(user));  
  
 User result = userService.getUserById(1L);  
 assertNotNull(result);  
 assertEquals("John Doe", result.getName());  
 }  
}

## Exercise 3: Mocking a Service Dependency in an Integration Test

import static org.mockito.Mockito.\*;  
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;  
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;  
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.jsonPath;  
import org.junit.jupiter.api.Test;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;  
import org.springframework.boot.test.context.SpringBootTest;  
import org.springframework.boot.test.mock.mockito.MockBean;  
import org.springframework.test.web.servlet.MockMvc;  
  
@SpringBootTest  
@AutoConfigureMockMvc  
public class UserIntegrationTest {  
  
 @Autowired  
 private MockMvc mockMvc;  
  
 @MockBean  
 private UserService userService;  
  
 @Test  
 public void testGetUserIntegration() throws Exception {  
 User user = new User();  
 user.setId(1L);  
 user.setName("Jane Smith");  
  
 when(userService.getUserById(1L)).thenReturn(user);  
  
 mockMvc.perform(get("/users/1"))  
 .andExpect(status().isOk())  
 .andExpect(jsonPath("$.name").value("Jane Smith"));  
 }  
}