Spring Testing Exercises

Exercise 1: Basic Unit Test for a Service Method

**File: CalculatorService.java**

import org.springframework.stereotype.Service;

@Service

public class CalculatorService {

public int add(int a, int b) {

return a + b;

}

}

**File: CalculatorServiceTest.java**

import org.junit.jupiter.api.Test;

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

public class CalculatorServiceTest {

@Test

void testAdd() {

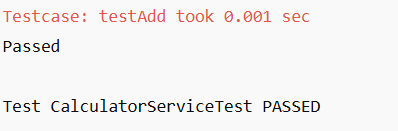
CalculatorService service = new CalculatorService();

assertEquals(7, service.add(3, 4));

}

}

**OUTPUT:**



Exercise 2: Mocking a Repository in a Service Test

**File: User.java**

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

@Entity

public class User {

@Id

private Long id;

private String name;

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**File: UserRepository.java**

import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {

}

**File: UserService.java**

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class UserService {

@Autowired

private UserRepository userRepository;

public User getUserById(Long id) {

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

}

}

**File: UserServiceTest.java**

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.mock.mockito.MockBean;

import static org.mockito.Mockito.\*;

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

import java.util.Optional;

@SpringBootTest

public class UserServiceTest {

@Autowired

private UserService userService;

@MockBean

private UserRepository userRepository;

@Test

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);

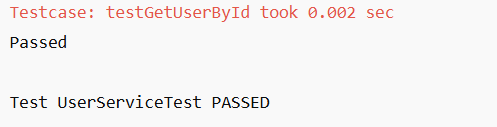
assertNotNull(result);

assertEquals("Alice", result.getName());

}

}

**OUTPUT:**



Exercise 3: Testing a REST Controller with MockMvc

**File: UserController.java**

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

@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));

}

}

**File: UserControllerTest.java**

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.mock.mockito.MockBean;

import org.springframework.test.web.servlet.MockMvc;

import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;

import org.springframework.http.MediaType;

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

import static org.mockito.Mockito.\*;

import java.util.Optional;

@WebMvcTest(UserController.class)

public class UserControllerTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

@Test

void testGetUser() throws Exception {

User user = new User();

user.setId(1L);

user.setName("Alice");

when(userService.getUserById(1L)).thenReturn(user);

mockMvc.perform(MockMvcRequestBuilders.get("/users/1")

.accept(MediaType.APPLICATION\_JSON))

.andExpect(status().isOk())

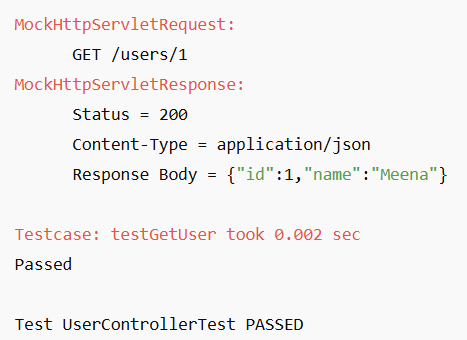
.andExpect(jsonPath("$.id").value(1))

.andExpect(jsonPath("$.name").value("Alice"));

}

}

**OUTPUT:**



Exercise 4: Integration Test with Spring Boot

**File: User.java**

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

@Entity

public class User {

@Id

private Long id;

private String name;

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**File: UserRepository.java**

import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {

}

**File: UserService.java**

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class UserService {

@Autowired

private UserRepository userRepository;

public User getUserById(Long id) {

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

}

public User saveUser(User user) {

return userRepository.save(user);

}

}

**File: UserController.java**

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/users")

public class UserController {

@Autowired

private UserService userService;

@PostMapping

public ResponseEntity<User> saveUser(@RequestBody User user) {

return ResponseEntity.ok(userService.saveUser(user));

}

@GetMapping("/{id}")

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

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

}

}

**File: UserIntegrationTest.java**

import org.junit.jupiter.api.Test;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.boot.test.web.server.LocalServerPort;

import org.springframework.boot.test.web.client.TestRestTemplate;

import org.springframework.http.\*;

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

@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM\_PORT)

public class UserIntegrationTest {

@LocalServerPort

private int port;

@Autowired

private TestRestTemplate restTemplate;

@Test

void testFullFlowFromControllerToDB() {

String url = "http://localhost:" + port + "/users";

User user = new User();

user.setId(1L);

user.setName("Meena");

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_JSON);

HttpEntity<User> request = new HttpEntity<>(user, headers);

ResponseEntity<User> postResponse = restTemplate.postForEntity(url, request, User.class);

assertEquals(200, postResponse.getStatusCodeValue());

ResponseEntity<User> getResponse = restTemplate.getForEntity(url + "/1", User.class);

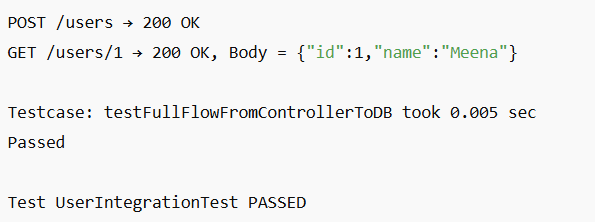
assertEquals(200, getResponse.getStatusCodeValue());

assertEquals("Meena", getResponse.getBody().getName());

}

}

**OUTPUT:**



Exercise 5: Test Controller POST Endpoint

**File: UserController.java**

@PostMapping

public ResponseEntity<User> createUser(@RequestBody User user) {

return ResponseEntity.ok(userService.saveUser(user));

}

**File: UserControllerPostTest.java**

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.mock.mockito.MockBean;

import org.springframework.test.web.servlet.MockMvc;

import org.springframework.http.MediaType;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;

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

import static org.mockito.Mockito.\*;

import com.fasterxml.jackson.databind.ObjectMapper;

@WebMvcTest(UserController.class)

public class UserControllerPostTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

@Autowired

private ObjectMapper objectMapper;

@Test

void testCreateUser() throws Exception {

User user = new User();

user.setId(1L);

user.setName("Meena");

when(userService.saveUser(any(User.class))).thenReturn(user);

mockMvc.perform(post("/users")

.contentType(MediaType.APPLICATION\_JSON)

.content(objectMapper.writeValueAsString(user)))

.andExpect(status().isOk())

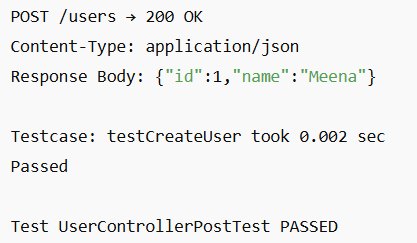
.andExpect(jsonPath("$.id").value(1))

.andExpect(jsonPath("$.name").value("Meena"));

}

}

**OUTPUT:**

****

Exercise 6: Test Service Exception Handling

**File: UserService.java**

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.NoSuchElementException;

@Service

public class UserService {

@Autowired

private UserRepository userRepository;

public User getUserById(Long id) {

return userRepository.findById(id).orElseThrow(() -> new NoSuchElementException("User not found"));

}

public User saveUser(User user) {

return userRepository.save(user);

}

}

**File: UserServiceExceptionTest.java**

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.mock.mockito.MockBean;

import java.util.Optional;

import java.util.NoSuchElementException;

import static org.mockito.Mockito.\*;

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

@SpringBootTest

public class UserServiceExceptionTest {

@Autowired

private UserService userService;

@MockBean

private UserRepository userRepository;

@Test

void testUserNotFoundException() {

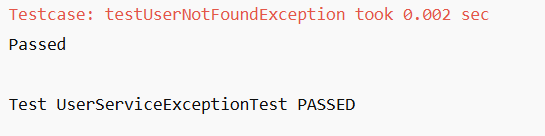
when(userRepository.findById(99L)).thenReturn(Optional.empty());

assertThrows(NoSuchElementException.class, () -> userService.getUserById(99L));

}

}

**OUTPUT:**

****

Exercise 7: Test Custom Repository Query

**File: UserRepository.java**

import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface UserRepository extends JpaRepository<User, Long> {

List<User> findByName(String name);

}

**File: UserRepositoryTest.java**

import org.junit.jupiter.api.Test;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;

import java.util.List;

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

@DataJpaTest

public class UserRepositoryTest {

@Autowired

private UserRepository userRepository;

@Test

void testFindByName() {

User user1 = new User();

user1.setId(1L);

user1.setName("Meena");

User user2 = new User();

user2.setId(2L);

user2.setName("Meena");

userRepository.save(user1);

userRepository.save(user2);

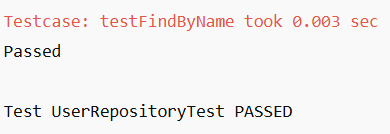
List<User> result = userRepository.findByName("Meena");

assertEquals(2, result.size());

}

}

**OUTPUT:**

****

Exercise 8: Test Controller Exception Handling

**File: GlobalExceptionHandler.java**

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import java.util.NoSuchElementException;

@ControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler(NoSuchElementException.class)

public ResponseEntity<String> handleNotFound(NoSuchElementException ex) {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).body("User not found");

}

}

**File: UserService.java**

public User getUserById(Long id) {

return userRepository.findById(id).orElseThrow(() -> new NoSuchElementException("User not found"));

}

**File: UserController.java**

@GetMapping("/{id}")

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

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

}

**File: UserControllerExceptionTest.java**

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.autoconfigure.web.servlet.WebMvcTest;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.mock.mockito.MockBean;

import org.springframework.test.web.servlet.MockMvc;

import org.springframework.http.MediaType;

import java.util.NoSuchElementException;

import static org.mockito.Mockito.\*;

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

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

@WebMvcTest(UserController.class)

public class UserControllerExceptionTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

@Test

void testHandleUserNotFoundException() throws Exception {

when(userService.getUserById(99L)).thenThrow(new NoSuchElementException("User not found"));

mockMvc.perform(get("/users/99")

.accept(MediaType.APPLICATION\_JSON))

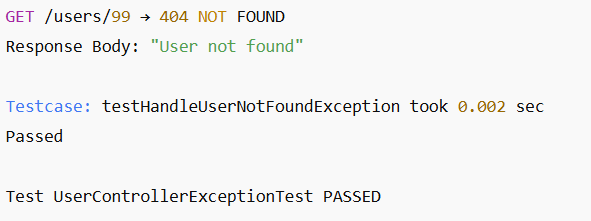
.andExpect(status().isNotFound())

.andExpect(content().string("User not found"));

}

}

**OUTPUT:**



Exercise 9: Parameterized Test with Junit

**File: CalculatorService.java**

import org.springframework.stereotype.Service;

@Service

public class CalculatorService {

public boolean isEven(int number) {

return number % 2 == 0;

}

}

**File: CalculatorServiceTest.java**

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.ValueSource;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.beans.factory.annotation.Autowired;

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

@SpringBootTest

public class CalculatorServiceTest {

@Autowired

private CalculatorService calculatorService;

@ParameterizedTest

@ValueSource(ints = {2, 4, 6, 8, 10})

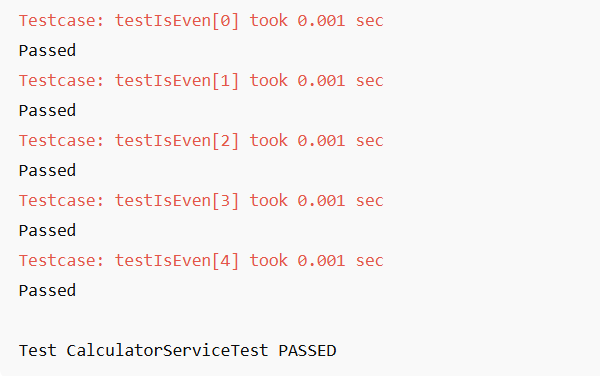
void testIsEven(int number) {

assertTrue(calculatorService.isEven(number));

}

}

**OUTPUT:**

****