<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-junit-jupiter</artifactId>

<scope>test</scope>

</dependency>

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

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

import java.util.List;

@RestController

@RequestMapping("/books")

public class BookController {

@Autowired

private BookService bookService;

@GetMapping

public List<Book> getAllBooks() {

return bookService.getAllBooks();

}

@PostMapping

public Book addBook(@RequestBody Book book) {

return bookService.addBook(book);

}

@GetMapping("/{id}")

public Book getBookById(@PathVariable Long id) {

return bookService.getBookById(id);

}

}

import static org.mockito.Mockito.\*;

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

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

import com.fasterxml.jackson.databind.ObjectMapper;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.MockitoAnnotations;

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

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

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

@WebMvcTest(BookController.class)

public class BookControllerTest {

@Autowired

private MockMvc mockMvc;

@Mock

private BookService bookService;

@InjectMocks

private BookController bookController;

@Autowired

private ObjectMapper objectMapper;

@BeforeEach

public void setup() {

MockitoAnnotations.openMocks(this);

}

@Test

public void testGetAllBooks() throws Exception {

// Arrange

Book book = new Book(1L, "Title", "Author");

when(bookService.getAllBooks()).thenReturn(List.of(book));

// Act and Assert

mockMvc.perform(get("/books"))

.andExpect(status().isOk())

.andExpect(content().contentType("application/json"))

.andExpect(jsonPath("$[0].id").value(1))

.andExpect(jsonPath("$[0].title").value("Title"))

.andExpect(jsonPath("$[0].author").value("Author"));

}

@Test

public void testAddBook() throws Exception {

// Arrange

Book book = new Book(1L, "Title", "Author");

when(bookService.addBook(any(Book.class))).thenReturn(book);

// Act and Assert

mockMvc.perform(post("/books")

.contentType("application/json")

.content(objectMapper.writeValueAsString(book)))

.andExpect(status().isOk())

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

.andExpect(jsonPath("$.title").value("Title"))

.andExpect(jsonPath("$.author").value("Author"));

}

@Test

public void testGetBookById() throws Exception {

// Arrange

Book book = new Book(1L, "Title", "Author");

when(bookService.getBookById(1L)).thenReturn(book);

// Act and Assert

mockMvc.perform(get("/books/1"))

.andExpect(status().isOk())

.andExpect(content().contentType("application/json"))

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

.andExpect(jsonPath("$.title").value("Title"))

.andExpect(jsonPath("$.author").value("Author"));

}

}