**Exercise 14: Online Bookstore - Integration Testing for REST Services**

1. **Set Up Spring Test for Integration Testing**

import org.junit.jupiter.api.Test;

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

@SpringBootTest

class BookstoreApplicationTests {

@Test

void contextLoads() {

// Context loads successfully

}

}

1. **MockMvc for End-to-End Testing**

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

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

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.http.MediaType;

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

@SpringBootTest

@AutoConfigureMockMvc

public class BookControllerIntegrationTest {

@Autowired

private MockMvc mockMvc;

@Test

public void testGetAllBooks() throws Exception {

// Perform the GET request and validate the response

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

.contentType(MediaType.APPLICATION\_JSON))

.andExpect(status().isOk())

.andExpect(jsonPath("$.size()").value(2)) // Assuming you have 2 books in the DB

.andExpect(jsonPath("$[0].title").value("Effective Java"));

}

}

1. **Database Integration with H2**

Add H2 Dependency to pom.xml:

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>test</scope>

</dependency>

Configure application-test.yml:

spring:

datasource:

url: jdbc:h2:mem:testdb

driver-class-name: org.h2.Driver

username: sa

password: password

h2:

console:

enabled: true

jpa:

hibernate:

ddl-auto: update

properties:

hibernate:

dialect: org.hibernate.dialect.H2Dialect

1. **Database Integration in Tests**

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

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

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.http.MediaType;

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

@SpringBootTest

@AutoConfigureMockMvc

public class BookControllerIntegrationTest {

@Autowired

private MockMvc mockMvc;

@Test

public void testCreateBook() throws Exception {

String bookJson = "{ \"title\": \"Domain-Driven Design\", \"author\": \"Eric Evans\", \"price\": 59.99 }";

mockMvc.perform(post("/api/books")

.content(bookJson)

.contentType(MediaType.APPLICATION\_JSON))

.andExpect(status().isCreated())

.andExpect(jsonPath("$.id").isNotEmpty())

.andExpect(jsonPath("$.title").value("Domain-Driven Design"));

}

}

1. **Ensure Test Isolation**

import org.junit.jupiter.api.AfterEach;

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

import org.springframework.jdbc.core.JdbcTemplate;

public class BookControllerIntegrationTest {

@Autowired

private JdbcTemplate jdbcTemplate;

@AfterEach

public void resetDb() {

jdbcTemplate.execute("DELETE FROM books");

}

}