**Day4 Labs: JUnit tests for UserController using @WebMvcTest + MockMvc**

**1) Maven test dependency (pom.xml)**

<dependency>

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

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

<scope>test</scope>

</dependency>

(That starter includes JUnit 5, Mockito, MockMvc, Jackson, etc.)

**2) User model**

package com.example.demo.model;

public class User {

private Long id;

private String name;

private String email;

public User() {}

public User(Long id, String name, String email) {

this.id = id; this.name = name; this.email = email;

}

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; }

public String getEmail() { return email; }

public void setEmail(String email) { this.email = email; }

// equals/hashCode/toString omitted for brevity

}

**3) UserService interface (to be mocked in tests)**

package com.example.demo.service;

import com.example.demo.model.User;

import java.util.\*;

public interface UserService {

List<User> findAll();

Optional<User> findById(Long id);

User save(User user);

Optional<User> update(Long id, User user);

boolean deleteById(Long id);

}

**4) UserController (REST endpoints)**

package com.example.demo.controller;

import com.example.demo.model.User;

import com.example.demo.service.UserService;

import org.springframework.http.ResponseEntity;

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

import org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import java.net.URI;

import java.util.List;

@RestController

@RequestMapping("/api/users")

public class UserController {

private final UserService userService;

public UserController(UserService userService) { this.userService = userService; }

@GetMapping

public ResponseEntity<List<User>> getAllUsers() {

return ResponseEntity.ok(userService.findAll());

}

@GetMapping("/{id}")

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

return userService.findById(id)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

@PostMapping

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

User saved = userService.save(user);

URI location = ServletUriComponentsBuilder.fromCurrentRequest()

.path("/{id}")

.buildAndExpand(saved.getId())

.toUri();

return ResponseEntity.created(location).body(saved);

}

@PutMapping("/{id}")

public ResponseEntity<User> updateUser(@PathVariable Long id, @RequestBody User user) {

return userService.update(id, user)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteUser(@PathVariable Long id) {

boolean deleted = userService.deleteById(id);

return deleted ? ResponseEntity.noContent().build() : ResponseEntity.notFound().build();

}

}

**5) UserControllerTest — full test class using @WebMvcTest + MockMvc**

package com.example.demo.controller;

import com.example.demo.model.User;

import com.example.demo.service.UserService;

import com.fasterxml.jackson.databind.ObjectMapper;

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.http.MediaType;

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

import java.util.\*;

import static org.hamcrest.Matchers.containsString;

import static org.mockito.ArgumentMatchers.\*;

import static org.mockito.Mockito.\*;

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

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

@WebMvcTest(UserController.class)

class UserControllerTest {

@Autowired

private MockMvc mockMvc;

@MockBean

private UserService userService;

@Test

void getAllUsers\_returnsList() throws Exception {

}

@Test

void getUserById\_found() throws Exception {

}

@Test

void getUserById\_notFound() throws Exception {

}

@Test

void createUser\_success() throws Exception {

}

@Test

void updateUser\_success() throws Exception {

}

@Test

void updateUser\_notFound() throws Exception {

}

@Test

void deleteUser\_success() throws Exception {

}

@Test

void deleteUser\_notFound() throws Exception {

}

}