**Step 1: Basic Testing with Controllers**  
package com.example.EmployeeManagementSystem.controller;

import com.example.entity.primary.Employee;

import com.example.entity.secondary.SomeOtherEntity;

import com.example.repository.primary.EmployeeRepository;

import com.example.repository.secondary.SomeOtherRepository;

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

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController

public class DataSourceTestController {

@Autowired

private EmployeeRepository employeeRepository;

@Autowired

private SomeOtherRepository someOtherRepository;

@GetMapping("/test-primary")

public List<Employee> testPrimaryDataSource() {

return employeeRepository.findAll();

}

@GetMapping("/test-secondary")

public List<SomeOtherEntity> testSecondaryDataSource() {

return someOtherRepository.findAll();

}

}

**Step 2: Create Unit Tests**

package com.example.EmployeeManagementSystem;

import com.example.entity.primary.Employee;

import com.example.repository.primary.EmployeeRepository;

import org.junit.jupiter.api.Test;

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

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

import static org.assertj.core.api.Assertions.assertThat;

@SpringBootTest

public class DataSourceConfigurationTest {

@Autowired

private EmployeeRepository employeeRepository;

@Test

public void testPrimaryDataSource() {

Employee employee = new Employee();

employee.setName("John Doe");

employee.setEmail("john.doe@example.com");

employeeRepository.save(employee);

assertThat(employeeRepository.findById(employee.getId())).isPresent();

}

}

**2. Implementing Pagination and Sorting**

**Step 1: Implement Pagination**package com.example.EmployeeManagementSystem.controller;

import com.example.entity.primary.Employee;

import com.example.repository.primary.EmployeeRepository;

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

import org.springframework.data.domain.Page;

import org.springframework.data.domain.PageRequest;

import org.springframework.data.domain.Pageable;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

@GetMapping("/employees")

public Page<Employee> getAllEmployees(

@RequestParam(defaultValue = "0") int page,

@RequestParam(defaultValue = "10") int size) {

Pageable pageable = PageRequest.of(page, size);

return employeeRepository.findAll(pageable);

}

}

**Step 2: Add Sorting Functionality**

package com.example.EmployeeManagementSystem.controller;

import com.example.entity.primary.Employee;

import com.example.repository.primary.EmployeeRepository;

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

import org.springframework.data.domain.Page;

import org.springframework.data.domain.PageRequest;

import org.springframework.data.domain.Pageable;

import org.springframework.data.domain.Sort;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

@GetMapping("/employees")

public Page<Employee> getAllEmployees(

@RequestParam(defaultValue = "0") int page,

@RequestParam(defaultValue = "10") int size,

@RequestParam(defaultValue = "name") String sortBy) {

Pageable pageable = PageRequest.of(page, size, Sort.by(sortBy));

return employeeRepository.findAll(pageable);

}

}

**Step 3: Combine Pagination and Sorting in Your Search Endpoint**

@GetMapping("/employees/search")

public Page<Employee> searchEmployees(

@RequestParam String name,

@RequestParam(defaultValue = "0") int page,

@RequestParam(defaultValue = "10") int size,

@RequestParam(defaultValue = "name") String sortBy) {

Pageable pageable = PageRequest.of(page, size, Sort.by(sortBy));

return employeeRepository.findByNameContaining(name, pageable);

}