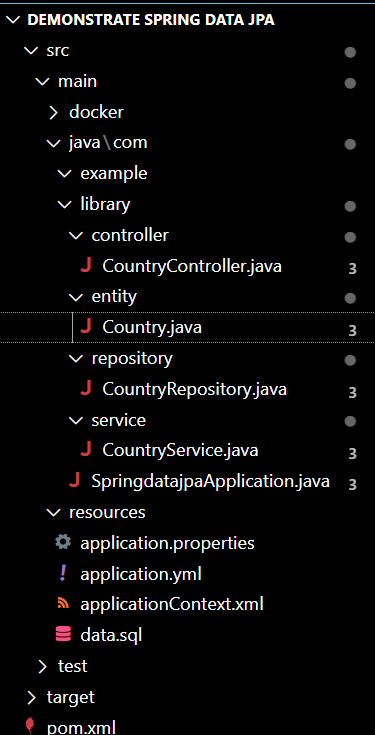
**Demonstrate implementation of Query Methods feature of Spring Data JPA**

Query Methods - Search by containing text, sorting, filter with starting text, fetch between dates, greater than or lesser than, top

Query methods - https://docs.spring.io/spring-data/jpa/docs/2.2.0.RELEASE/reference/html/#jpa.query-methods.query-creation

**STRUCTURE:**



**CountryController.java**

package com.example.library.controller;

import com.example.library.entity.Country;

import com.example.library.service.CountryService;

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

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

import java.time.LocalDate;

import java.util.List;

@RestController

@RequestMapping("/countries")

public class CountryController {

    @Autowired

    private CountryService service;

    @PostMapping

    public Country addCountry(@RequestBody Country country) {

        return service.addCountry(country);

    }

    @GetMapping

    public List<Country> getAllCountries() {

        return service.getAllCountries();

    }

    @GetMapping("/search")

    public List<Country> searchByName(@RequestParam String name) {

        return service.searchByName(name);

    }

    @GetMapping("/starts-with")

    public List<Country> searchByPrefix(@RequestParam String prefix) {

        return service.searchByPrefix(prefix);

    }

    @GetMapping("/between-dates")

    public List<Country> getBetweenDates(@RequestParam String start, @RequestParam String end) {

        return service.getBetweenDates(LocalDate.parse(start), LocalDate.parse(end));

    }

    @GetMapping("/greater-than")

    public List<Country> getGreaterThanPopulation(@RequestParam long population) {

        return service.getGreaterThanPopulation(population);

    }

    @GetMapping("/less-than")

    public List<Country> getLessThanPopulation(@RequestParam long population) {

        return service.getLessThanPopulation(population);

    }

    @GetMapping("/top3")

    public List<Country> getTop3Populous() {

        return service.getTop3Populous();

    }

    @GetMapping("/sorted")

    public List<Country> getSortedByName() {

        return service.getSortedByName();

    }

}

**Country.java**

package com.example.library.controller;

import com.example.library.entity.Country;

import com.example.library.service.CountryService;

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

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

import java.time.LocalDate;

import java.util.List;

@RestController

@RequestMapping("/countries")

public class CountryController {

    @Autowired

    private CountryService service;

    @PostMapping

    public Country addCountry(@RequestBody Country country) {

        return service.addCountry(country);

    }

    @GetMapping

    public List<Country> getAllCountries() {

        return service.getAllCountries();

    }

    @GetMapping("/search")

    public List<Country> searchByName(@RequestParam String name) {

        return service.searchByName(name);

    }

    @GetMapping("/starts-with")

    public List<Country> searchByPrefix(@RequestParam String prefix) {

        return service.searchByPrefix(prefix);

    }

    @GetMapping("/between-dates")

    public List<Country> getBetweenDates(@RequestParam String start, @RequestParam String end) {

        return service.getBetweenDates(LocalDate.parse(start), LocalDate.parse(end));

    }

    @GetMapping("/greater-than")

    public List<Country> getGreaterThanPopulation(@RequestParam long population) {

        return service.getGreaterThanPopulation(population);

    }

    @GetMapping("/less-than")

    public List<Country> getLessThanPopulation(@RequestParam long population) {

        return service.getLessThanPopulation(population);

    }

    @GetMapping("/top3")

    public List<Country> getTop3Populous() {

        return service.getTop3Populous();

    }

    @GetMapping("/sorted")

    public List<Country> getSortedByName() {

        return service.getSortedByName();

    }

}

**CountryRepository.java**

package com.example.library.repository;

import java.time.LocalDate;

import java.util.List;

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

import com.example.library.entity.Country;

public interface CountryRepository extends JpaRepository<Country, Long> {

    List<Country> findByNameContaining(String name); // contains text

    List<Country> findByNameStartingWith(String prefix); // starts with

    List<Country> findByIndependenceDateBetween(LocalDate start, LocalDate end); // between dates

    List<Country> findByPopulationGreaterThan(long population); // population greater than

    List<Country> findByPopulationLessThan(long population); // population less than

    List<Country> findTop3ByOrderByPopulationDesc(); // top 3 populous

    List<Country> findAllByOrderByNameAsc(); // sorted by name

}

**CountryService.java**

package com.example.library.service;

import java.time.LocalDate;

import java.util.List;

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

import org.springframework.stereotype.Service;

import com.example.library.entity.Country;

import com.example.library.repository.CountryRepository;

@Service

public class CountryService {

    @Autowired

    private CountryRepository repository;

    public List<Country> getAllCountries() {

        return repository.findAll();

    }

    public Country addCountry(Country country) {

        return repository.save(country);

    }

    public List<Country> searchByName(String text) {

        return repository.findByNameContaining(text);

    }

    public List<Country> searchByPrefix(String prefix) {

        return repository.findByNameStartingWith(prefix);

    }

    public List<Country> getBetweenDates(LocalDate from, LocalDate to) {

        return repository.findByIndependenceDateBetween(from, to);

    }

    public List<Country> getGreaterThanPopulation(long value) {

        return repository.findByPopulationGreaterThan(value);

    }

    public List<Country> getLessThanPopulation(long value) {

        return repository.findByPopulationLessThan(value);

    }

    public List<Country> getTop3Populous() {

        return repository.findTop3ByOrderByPopulationDesc();

    }

    public List<Country> getSortedByName() {

        return repository.findAllByOrderByNameAsc();

    }

}

**SpringdatajpaApplication.java**

package com.example.library;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringdatajpaApplication {

    public static void main(String[] args) {

        SpringApplication.run(SpringdatajpaApplication.class, args);

    }

}

**Application.properties**

# ============ DATABASE SETTINGS ============

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

# ============ JPA SETTINGS ============

spring.jpa.hibernate.ddl-auto=create

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

# ============ H2 CONSOLE ============

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

# ============ DATA.SQL EXECUTION ORDER ============

# This ensures data.sql runs AFTER Hibernate creates tables

spring.datasource.initialization-mode=always

spring.jpa.defer-datasource-initialization=true

server.port=8084

**applicationContext.xml**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

       xsi:schemaLocation="

           http://www.springframework.org/schema/beans

           http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- BookRepository bean -->

    <bean id="bookRepository" class="com.example.library.repository.BookRepository" />

    <!-- BookService bean with constructor and setter injection -->

    <bean id="bookService" class="com.example.library.service.BookService">

        <!-- Constructor injection -->

        <constructor-arg value="Library Book Service" />

        <!-- Setter injection -->

        <property name="bookRepository" ref="bookRepository" />

    </bean>

</beans>

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

         http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>

    <artifactId>springdatajpa</artifactId>

    <version>1.0.0</version>

    <packaging>jar</packaging>

    <name>Spring Data JPA</name>

    <description>Country Code Demo</description>

    <parent>

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

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

        <version>3.2.0</version>

        <relativePath/> <!-- lookup parent from repository -->

    </parent>

    <properties>

        <java.version>17</java.version>

    </properties>

    <dependencies>

        <!-- Spring Boot Starter Web -->

        <dependency>

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

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

        </dependency>

        <!-- Spring Boot Starter Data JPA -->

        <dependency>

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

            <artifactId>spring-boot-starter-data-jpa</artifactId>

        </dependency>

        <!-- H2 Database -->

        <dependency>

            <groupId>com.h2database</groupId>

            <artifactId>h2</artifactId>

            <scope>runtime</scope>

        </dependency>

        <!-- Lombok -->

        <dependency>

            <groupId>org.projectlombok</groupId>

            <artifactId>lombok</artifactId>

            <optional>true</optional>

        </dependency>

        <!-- Devtools -->

        <dependency>

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

            <artifactId>spring-boot-devtools</artifactId>

            <scope>runtime</scope>

        </dependency>

        <!-- Testing -->

        <dependency>

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

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

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <!-- Spring Boot Maven Plugin -->

            <plugin>

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

                <artifactId>spring-boot-maven-plugin</artifactId>

            </plugin>

        </plugins>

    </build>

</project>

**Data.sql**

INSERT INTO country (name, code, population, independence\_date) VALUES

('India', 'IN', 1400000000, '1947-08-15'),

('United States', 'US', 330000000, '1776-07-04'),

('Japan', 'JP', 125000000, '1952-04-28'),

('Brazil', 'BR', 211000000, '1822-09-07'),

('Australia', 'AU', 25000000, '1901-01-01'),

('Canada', 'CA', 38000000, '1867-07-01');

**OUTPUT:**

