Demonstrate implementation of Query Methods feature of Spring Data JPA

QueryMethodsDemoApplication.java

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class QueryMethodsDemoApplication {

public static void main(String[] args) {

SpringApplication.run(QueryMethodsDemoApplication.class, args);

}

}

Book.java

package com.example.demo.entity;

import jakarta.persistence.\*;

import java.math.BigDecimal;

import java.time.LocalDate;

@Entity

@Table(name = "books")

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

private BigDecimal price;

private LocalDate publishedDate;

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public String getTitle() { return title; }

public void setTitle(String title) { this.title = title; }

public String getAuthor() { return author; }

public void setAuthor(String author) { this.author = author; }

public BigDecimal getPrice() { return price; }

public void setPrice(BigDecimal price) { this.price = price; }

public LocalDate getPublishedDate() { return publishedDate; }

public void setPublishedDate(LocalDate publishedDate) { this.publishedDate = publishedDate; }

}

BookRepository.java

package com.example.demo.repository;

import com.example.demo.entity.Book;

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

import java.math.BigDecimal;

import java.time.LocalDate;

import java.util.List;

public interface BookRepository extends JpaRepository<Book, Long> {

List<Book> findByTitleContaining(String text);

List<Book> findByAuthorStartingWith(String prefix);

List<Book> findByTitleContainingOrderByPriceAsc(String text);

List<Book> findByPublishedDateBetween(LocalDate start, LocalDate end);

List<Book> findByPriceGreaterThan(BigDecimal price);

List<Book> findByPriceLessThanEqual(BigDecimal price);

List<Book> findTop3ByOrderByPriceDesc();

}

BookService.java

package com.example.demo.service;

import com.example.demo.entity.Book;

import com.example.demo.repository.BookRepository;

import org.springframework.stereotype.Service;

import java.math.BigDecimal;

import java.time.LocalDate;

import java.util.List;

@Service

public class BookService {

private final BookRepository repo;

public BookService(BookRepository repo) { this.repo = repo; }

public Book save(Book b) { return repo.save(b); }

public List<Book> findAll() { return repo.findAll(); }

public List<Book> searchTitle(String text) { return repo.findByTitleContaining(text); }

public List<Book> searchAuthorPrefix(String prefix) { return repo.findByAuthorStartingWith(prefix); }

public List<Book> searchTitleSortPrice(String text) { return repo.findByTitleContainingOrderByPriceAsc(text); }

public List<Book> betweenDates(LocalDate start, LocalDate end) { return repo.findByPublishedDateBetween(start, end); }

public List<Book> priceGreater(BigDecimal price) { return repo.findByPriceGreaterThan(price); }

public List<Book> priceLessOrEqual(BigDecimal price) { return repo.findByPriceLessThanEqual(price); }

public List<Book> topExpensive() { return repo.findTop3ByOrderByPriceDesc(); }

}

BookController.java

package com.example.demo.controller;

import com.example.demo.entity.Book;

import com.example.demo.service.BookService;

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

import java.math.BigDecimal;

import java.time.LocalDate;

import java.util.List;

@RestController

@RequestMapping("/books")

public class BookController {

private final BookService svc;

public BookController(BookService svc) { this.svc = svc; }

@PostMapping

public Book add(@RequestBody Book book) { return svc.save(book); }

@GetMapping

public List<Book> all() { return svc.findAll(); }

@GetMapping("/title")

public List<Book> byTitle(@RequestParam String q) { return svc.searchTitle(q); }

@GetMapping("/author")

public List<Book> byAuthorPrefix(@RequestParam String prefix) { return svc.searchAuthorPrefix(prefix); }

@GetMapping("/title-price")

public List<Book> byTitleSortPrice(@RequestParam String q) { return svc.searchTitleSortPrice(q); }

@GetMapping("/between")

public List<Book> byDates(@RequestParam String from, @RequestParam String to) {

return svc.betweenDates(LocalDate.parse(from), LocalDate.parse(to));

}

@GetMapping("/price/gt")

public List<Book> priceGt(@RequestParam BigDecimal price) { return svc.priceGreater(price); }

@GetMapping("/price/lte")

public List<Book> priceLte(@RequestParam BigDecimal price) { return svc.priceLessOrEqual(price); }

@GetMapping("/top")

public List<Book> top() { return svc.topExpensive(); }

}

application.properties

spring.datasource.url=jdbc:mysql://localhost:3306/librarybd2

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

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/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>query-methods-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<parent>

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

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

<version>3.3.0</version>

</parent>

<properties>

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

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**Output:**







