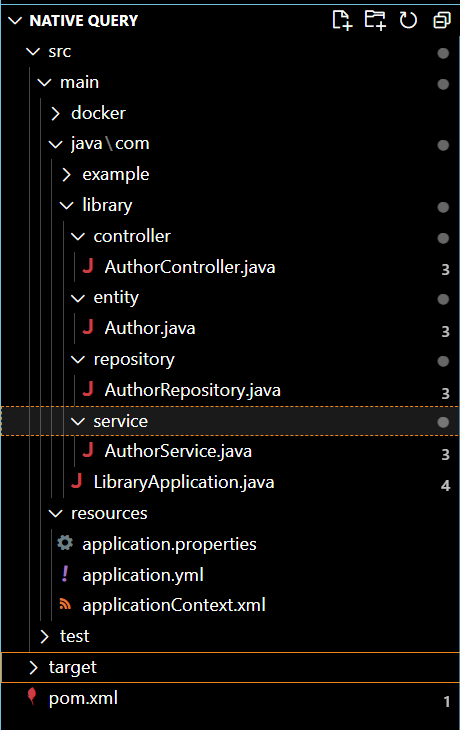
**Demonstrate writing Hibernate Query Language and Native Query**

HQL stands for Hibernate Query Language, JPQL stands for Java Persistence Query Language, Compare HQL and JPQL, @Query annotation, HQL fetch keyword, aggregate functions in HQL, Native Query, nativeQuery attribute

Reference - https://docs.jboss.org/hibernate/orm/4.3/devguide/en-US/html/ch11.html

Features of JPA Query - <https://www.baeldung.com/spring-data-jpa-query>

**STRUCTURE:**



**AuthorController.java**

package com.example.library.controller;

import com.example.library.entity.Author;

import com.example.library.repository.AuthorRepository;

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

import java.util.List;

@RestController

@RequestMapping("/authors")

public class AuthorController {

    private final AuthorRepository repository;

    public AuthorController(AuthorRepository repository) {

        this.repository = repository;

    }

    @GetMapping("/hql")

    public List<Author> getAuthorsByHQL() {

        return repository.findAuthorsByHQL("British");

    }

    @GetMapping("/native")

    public List<Author> getAuthorsByNative() {

        return repository.findAuthorsByNative("British");

    }

}

**Author.java**

package com.example.library.entity;

import jakarta.persistence.\*;

@Entity

public class Author {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String name;

    private String nationality;

    public Author() {}

    public Author(Long id, String name, String nationality) {

        this.id = id;

        this.name = name;

        this.nationality = nationality;

    }

    // Getters and setters

    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 getNationality() { return nationality; }

    public void setNationality(String nationality) { this.nationality = nationality; }

}

**AuthorRepository.java**

package com.example.library.repository;

import com.example.library.entity.Author;

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

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

import org.springframework.stereotype.Repository;

import org.springframework.data.repository.query.Param;

import java.util.List;

@Repository

public interface AuthorRepository extends JpaRepository<Author, Long> {

    @Query("SELECT a FROM Author a WHERE a.nationality = :nationality")

    List<Author> findAuthorsByHQL(@Param("nationality") String nationality);

    @Query(value = "SELECT \* FROM author WHERE nationality = :nationality", nativeQuery = true)

    List<Author> findAuthorsByNative(@Param("nationality") String nationality);

}

**AuthorService.java**

package com.example.library.service;

import com.example.library.entity.Author;

import com.example.library.repository.AuthorRepository;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class AuthorService {

    private final AuthorRepository authorRepository;

    public AuthorService(AuthorRepository authorRepository) {

        this.authorRepository = authorRepository;

    }

    public List<Author> getAuthorsByHQL(String nationality) {

        return authorRepository.findAuthorsByHQL(nationality);

    }

    public List<Author> getAuthorsByNative(String nationality) {

        return authorRepository.findAuthorsByNative(nationality);

    }

}

**LibraryApplication.java**

package com.example.library;

import com.example.library.entity.Author;

import com.example.library.repository.AuthorRepository;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

@SpringBootApplication

public class LibraryApplication {

    public static void main(String[] args) {

        SpringApplication.run(LibraryApplication.class, args);

    }

    @Bean

    CommandLineRunner runner(AuthorRepository authorRepository) {

        return args -> {

            authorRepository.save(new Author(null, "J.K. Rowling", "British"));

            authorRepository.save(new Author(null, "George Orwell", "British"));

            authorRepository.save(new Author(null, "Mark Twain", "American"));

        };

    }

}

**Application.properties**

server.port=8085

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

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

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update

spring.h2.console.enabled=true

**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 Web -->

        <dependency>

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

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

        </dependency>

        <!-- Spring 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>

            <version>1.18.30</version>

            <scope>provided</scope>

        </dependency>

        <!-- DevTools (Hot Reloading) -->

        <dependency>

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

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

            <scope>runtime</scope>

        </dependency>

        <!-- Spring Boot Test -->

        <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>

**OUTPUT:**

