Lab Exercise 03: Building a Spring Data Microservice

📌 Lab Title: Integrating Spring Data JPA in a Microservice

🧩 Scenario:

You are developing a microservice for an online store that manages product data. You will create a Spring Boot microservice that uses Spring Data JPA to persist and retrieve `Product` entities from an Oracle Database 23ai.

🛠️ Lab Requirements

- Java 17
- Spring Boot 3.x
- Maven
- Oracle Database 23ai
- HikariCP
- Flyway
- Spring Data JPA

📋 Lab Steps

Step 1: Initialize the Spring Boot Project

Use Spring Initializr to generate a project with the following dependencies:

- Spring Web
- Spring Data JPA
- Flyway Migration
- Oracle Driver
- HikariCP

Save the project as `spring-data-lab`.

Step 2: Configure `application.properties`

Create the following configuration in `src/main/resources/application.properties`:

```
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:XE
spring.datasource.username=product_user
spring.datasource.password=yourpassword
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver
spring.datasource.hikari.maximum-pool-size=5
```

```
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.OracleDialect

flyway.enabled=true
flyway.locations=classpath:db/migration
```

Step 3: Create the Database Schema Using Flyway

Create the following migration file:

File: `src/main/resources/db/migration/V1__create_products_table.sql`

```
-- Create the products table
CREATE TABLE products (
    id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    product_name VARCHAR2(255) NOT NULL,
    price NUMBER(10,2) NOT NULL
);
```

Step 4: Define the `Product` Entity

Create the `Product.java` entity class:

File: `src/main/java/com/example/springdatalab/model/Product.java`

```
package com.example.springdatalab.model;

import jakarta.persistence.*;

@Entity
@Table(name = "products")
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "product_name", nullable = false)
    private String name;

    private Double price;

    public Product() {}

    public Product(String name, Double price) {
        this.name = name;
        this.price = price;
    }
```

```
    // Getters and setters
}
```

Step 5: Create the Repository Interface

File: `src/main/java/com/example/springdatalab/repository/ProductRepository.java`

```
package com.example.springdatalab.repository;

import com.example.springdatalab.model.Product;
import org.springframework.data.jpa.repository.JpaRepository;

public interface ProductRepository extends JpaRepository<Product, Long> {
    Product findByName(String name);
}
```

Step 6: Create a Service Layer

File: `src/main/java/com/example/springdatalab/service/ProductService.java`

```
package com.example.springdatalab.service;

import com.example.springdatalab.model.Product;
import com.example.springdatalab.repository.ProductRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class ProductService {

    @Autowired
    private ProductRepository productRepository;

    public List<Product> getAllProducts() {
        return productRepository.findAll();
    }

    public Product getProductByName(String name) {
        return productRepository.findByName(name);
    }

    public Product saveProduct(Product product) {
        return productRepository.save(product);
    }
}
```

Step 7: Expose a REST Controller

File: `src/main/java/com/example/springdatalab/controller/ProductController.java`

```
package com.example.springdatalab.controller;

import com.example.springdatalab.model.Product;
import com.example.springdatalab.service.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/products")
public class ProductController {

    @Autowired
    private ProductService productService;

    @GetMapping
    public List<Product> getAllProducts() {
        return productService.getAllProducts();
    }

    @GetMapping("/{name}")
    public Product getProduct(@PathVariable String name) {
        return productService.getProductByName(name);
    }

    @PostMapping
    public Product createProduct(@RequestBody Product product) {
        return productService.saveProduct(product);
    }
}
```

Step 8: Test the Application

1. Start Oracle Database 23ai and create the `product_user` schema:

```
-- Connect as SYSTEM or SYS
CREATE USER product_user IDENTIFIED BY yourpassword;
GRANT CONNECT, RESOURCE, CREATE SESSION TO product_user;
GRANT CREATE TABLE TO product_user;
GRANT UNLIMITED TABLESPACE TO product_user;
```

2. Run the Spring Boot application using:

```
./mvnw spring-boot:run
```

3. Use `curl` or Postman to test the endpoints:

Add a product:

```
curl -X POST http://localhost:8080/api/products -H "Content-Type:
application/json" -d '{"name":"Laptop","price":1200.00}'
```

Get all products:

```
curl http://localhost:8080/api/products
```

Get product by name:

```
curl http://localhost:8080/api/products/Laptop
```
