**Exercise 8: Online Bookstore - Implementing CRUD Operations**

**Business Scenario:**

Implement Create, Read, Update, and Delete operations for the Book and Customer entities.

**Instructions:**

1. **CRUD Endpoints:**
   * Implement endpoints for creating, reading, updating, and deleting books and customers.
2. **Validating Input Data:**
   * Use validation annotations like **@NotNull, @Size**, and **@Min** to validate input data.
3. **Optimistic Locking:**

Implement optimistic locking for concurrent updates using JPA versioning

**Book.java**

package com.example.bookstoreapi.model;

import jakarta.persistence.\*;

import jakarta.validation.constraints.Min;

import jakarta.validation.constraints.NotNull;

import jakarta.validation.constraints.Size;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@NotNull(message = "Title is mandatory")

@Size(min = 2, max = 100, message = "Title must be between 2 and 100 characters")

private String title;

@NotNull(message = "Author is mandatory")

@Size(min = 2, max = 50, message = "Author must be between 2 and 50 characters")

private String author;

@NotNull(message = "Price is mandatory")

@Min(value = 0, message = "Price must be greater than or equal to 0")

private Double price;

@Version

private int version;

// Constructors

public Book() {}

public Book(Long id, String title, String author, Double price) {

this.id = id;

this.title = title;

this.author = author;

this.price = price;

}

// Getters and Setters

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 Double getPrice() {

return price;

}

public void setPrice(Double price) {

this.price = price;

}

public int getVersion() {

return version;

}

public void setVersion(int version) {

this.version = version;

}

}

**Customer.java**

package com.example.bookstoreapi.model;

import jakarta.persistence.\*;

import jakarta.validation.constraints.Email;

import jakarta.validation.constraints.NotNull;

import jakarta.validation.constraints.Size;

@Entity

public class Customer {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@NotNull(message = "Name is mandatory")

@Size(min = 2, max = 50, message = "Name must be between 2 and 50 characters")

private String name;

@NotNull(message = "Email is mandatory")

@Email(message = "Email should be valid")

private String email;

@Version

private int version;

// Constructors

public Customer() {}

public Customer(Long id, String name, String email) {

this.id = id;

this.name = name;

this.email = email;

}

// 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 getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public int getVersion() {

return version;

}

public void setVersion(int version) {

this.version = version;

}

}

**BookRepository.java**

package com.example.bookstoreapi.repository;

import com.example.bookstoreapi.model.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

}

**CustomerRepository.java**

package com.example.bookstoreapi.repository;

import com.example.bookstoreapi.model.Customer;

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

public interface CustomerRepository extends JpaRepository<Customer, Long> {

}

**BookService.java**

package com.example.bookstoreapi.service;

import com.example.bookstoreapi.model.Book;

import com.example.bookstoreapi.repository.BookRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.Optional;

@Service

public class BookService {

@Autowired

private BookRepository bookRepository;

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

public Optional<Book> getBookById(Long id) {

return bookRepository.findById(id);

}

public Book saveBook(Book book) {

return bookRepository.save(book);

}

public void deleteBook(Long id) {

bookRepository.deleteById(id);

}

}

**CustomerService.java**

package com.example.bookstoreapi.service;

import com.example.bookstoreapi.model.Customer;

import com.example.bookstoreapi.repository.CustomerRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.Optional;

@Service

public class CustomerService {

@Autowired

private CustomerRepository customerRepository;

public List<Customer> getAllCustomers() {

return customerRepository.findAll();

}

public Optional<Customer> getCustomerById(Long id) {

return customerRepository.findById(id);

}

public Customer saveCustomer(Customer customer) {

return customerRepository.save(customer);

}

public void deleteCustomer(Long id) {

customerRepository.deleteById(id);

}

}

**BookController.java**

package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.model.Book;

import com.example.bookstoreapi.service.BookService;

import jakarta.validation.Valid;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/api/books")

public class BookController {

@Autowired

private BookService bookService;

@GetMapping

public List<Book> getAllBooks() {

return bookService.getAllBooks();

}

@GetMapping("/{id}")

public ResponseEntity<Book> getBookById(@PathVariable Long id) {

Optional<Book> book = bookService.getBookById(id);

return book.map(ResponseEntity::ok)

.orElseGet(() -> ResponseEntity.notFound().build());

}

@PostMapping

public ResponseEntity<Book> createBook(@Valid @RequestBody Book book) {

Book createdBook = bookService.saveBook(book);

return new ResponseEntity<>(createdBook, HttpStatus.CREATED);

}

@PutMapping("/{id}")

public ResponseEntity<Book> updateBook(@PathVariable Long id, @Valid @RequestBody Book bookDetails) {

Optional<Book> optionalBook = bookService.getBookById(id);

if (optionalBook.isPresent()) {

Book book = optionalBook.get();

book.setTitle(bookDetails.getTitle());

book.setAuthor(bookDetails.getAuthor());

book.setPrice(bookDetails.getPrice());

Book updatedBook = bookService.saveBook(book);

return ResponseEntity.ok(updatedBook);

} else {

return ResponseEntity.notFound().build();

}

}

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteBook(@PathVariable Long id) {

if (bookService.getBookById(id).isPresent()) {

bookService.deleteBook(id);

return ResponseEntity.noContent().build();

} else {

return ResponseEntity.notFound().build();

}

}

}

**CustomerController.java**

package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.model.Customer;

import com.example.bookstoreapi.service.CustomerService;

import jakarta.validation.Valid;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/api/customers")

public class CustomerController {

@Autowired

private CustomerService customerService;

@GetMapping

public List<Customer> getAllCustomers() {

return customerService.getAllCustomers();

}

@GetMapping("/{id}")

public ResponseEntity<Customer> getCustomerById(@PathVariable Long id) {

Optional<Customer> customer = customerService.getCustomerById(id);

return customer.map(ResponseEntity::ok)

.orElseGet(() -> ResponseEntity.notFound().build());

}

@PostMapping

public ResponseEntity<Customer> createCustomer(@Valid @RequestBody Customer customer) {

Customer createdCustomer = customerService.saveCustomer(customer);

return new ResponseEntity<>(createdCustomer, HttpStatus.CREATED);

}

@PutMapping("/{id}")

public ResponseEntity<Customer> updateCustomer(@PathVariable Long id, @Valid @RequestBody Customer customerDetails) {

Optional<Customer> optionalCustomer = customerService.getCustomerById(id);

if (optionalCustomer.isPresent()) {

Customer customer = optionalCustomer.get();

customer.setName(customerDetails.getName());

customer.setEmail(customerDetails.getEmail());

Customer updatedCustomer = customerService.saveCustomer(customer);

return ResponseEntity.ok(updatedCustomer);

} else {

return ResponseEntity.notFound().build();

}

}

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteCustomer(@PathVariable Long id) {

if (customerService.getCustomerById(id).isPresent()) {

customerService.deleteCustomer(id);

return ResponseEntity.noContent().build();

} else {

return ResponseEntity.notFound().build();

}

}

}

**Configuration**

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.hibernate.ddl-auto=update