Task: 11

Hibernate Standalone and web Application

Date:

Develop an Online Book Store web application using hibernate with n-tier architecture. The application should be able to add and search all books such as online-purchase/paperback, Indian author / Foreign Author and High price/Low price.

- Add new books
- Update book information
- Delete book
- Show all books
- Search all books.

Algorithm

1. Create database `online_bookstore` and `books` table.
2. Build a Maven web project with Spring MVC (controller + JSP) and Hibernate.
3. Configure `hibernate.cfg.xml` for DB and mapping; configure `web.xml` and Spring `dispatcher-servlet.xml`.
4. Create `Book` entity annotated for Hibernate.
5. Implement `HibernateUtil` (SessionFactory).
6. Implement `BookDAO` (CRUD & search methods).
7. Implement `BookService` that calls DAO and implements business rules.
8. Implement `BookController` (Spring MVC) with endpoints:
   - `/home` — homepage
   - `/books` — list all books
   - `/book/add` GET/POST — add book
   - `/book/edit/{id}` GET/POST — update book
   - `/book/delete/{id}` — delete book
   - `/book/search` — search/filter books
9. Create JSP views: `home.jsp`, `listBooks.jsp`, `addBook.jsp`, `editBook.jsp`, `searchResults.jsp`.
10. Deploy to Tomcat, run, and test flows (add, update, delete, list, search).
    - Homepage with navigation link to grocery list.
    - Product list displayed in table form.

Program:

DATABASE (MySQL)

```
CREATE DATABASE online_bookstore;
USE online_bookstore;
CREATE TABLE books (
 id INT AUTO_INCREMENT PRIMARY KEY,
 title VARCHAR(100),
 author VARCHAR(100),
 price DOUBLE
);
```

HIBERNATE CONFIG (hibernate.cfg.xml)

```
<hibernate-configuration>
 <session-factory>
  <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
  <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/online_bookstore</property>
  <property name="hibernate.connection.username">root</property>
  <property name="hibernate.connection.password">root</property>
  <property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>
  <property name="hibernate.hbm2ddl.auto">update</property>
  <mapping class="Book"/>
 </session-factory>
</hibernate-configuration>
```

Book.java (Entity)

```
import javax.persistence.*;

@Entity
@Table(name="books")
public class Book {
 @Id @GeneratedValue(strategy=GenerationType.IDENTITY)
 private int id;
 private String title;
 private String author;
 private double price;

 public Book() {}
 public Book(String t,String a,double p){title=t;author=a;price=p;}

 // getters, setters, toString
 public String toString(){return id+" "+title+" "+author+" Rs."+price;}
}
```

HibernateUtil.java

```
import org.hibernate.*;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {
 private static SessionFactory factory = new
Configuration().configure().buildSessionFactory();
 public static SessionFactory getFactory(){ return factory; }
}
```

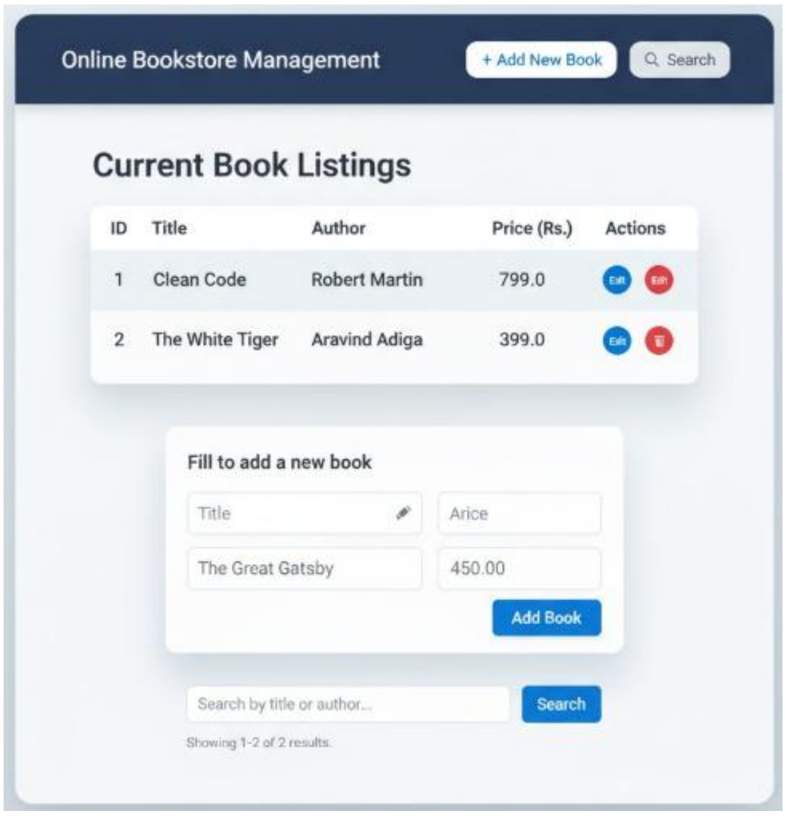
<u>BookDAO.java (CRUD + search)</u>

```
import org.hibernate.*;
import java.util.*;

public class BookDAO {
 public void add(Book b){
  Session s=HibernateUtil.getFactory().openSession();
  s.beginTransaction(); s.save(b); s.getTransaction().commit(); s.close();
 }
 public void update(Book b){
  Session s=HibernateUtil.getFactory().openSession();
  s.beginTransaction(); s.update(b); s.getTransaction().commit(); s.close();
 }
 public void delete(int id){
  Session s=HibernateUtil.getFactory().openSession();
  Book b=s.get(Book.class,id);
  if(b!=null){ s.beginTransaction(); s.delete(b); s.getTransaction().commit();}
  s.close();
 }
 public List<Book> showAll(){
  Session s=HibernateUtil.getFactory().openSession();
  List<Book> list=s.createQuery("from Book",Book.class).list();
  s.close(); return list;
 }
 public List<Book> search(String key){
  Session s=HibernateUtil.getFactory().openSession();
  List<Book> list=s.createQuery("from Book where title like :k or author like :k",Book.class)
                  .setParameter("k","%"+key+"%").list();
  s.close(); return list;
 }
}
```

MainApp.java (Simple Console Controller)

```
import java.util.*;

public class MainApp {
 public static void main(String[] args){
  Scanner sc=new Scanner(System.in);
  BookDAO dao=new BookDAO();
  while(true){
   System.out.println("\n1.Add 2.Update 3.Delete 4.ShowAll 5.Search 6.Exit");
   int ch=sc.nextInt(); sc.nextLine();
   switch(ch){
    case 1:
      System.out.print("Title: "); String t=sc.nextLine();
      System.out.print("Author: "); String a=sc.nextLine();
      System.out.print("Price: "); double p=sc.nextDouble();
      dao.add(new Book(t,a,p)); break;
    case 2:
      System.out.print("Enter ID to update: "); int id=sc.nextInt(); sc.nextLine();
      System.out.print("New title: "); t=sc.nextLine();
      System.out.print("New author: "); a=sc.nextLine();
      System.out.print("New price: "); p=sc.nextDouble();
      Book b=new Book(t,a,p); b.setId(id); dao.update(b); break;
    case 3:
      System.out.print("ID to delete: "); dao.delete(sc.nextInt()); break;
    case 4:
      dao.showAll().forEach(System.out::println); break;
    case 5:
      System.out.print("Search keyword: "); String key=sc.next();
      dao.search(key).forEach(System.out::println); break;
    case 6:
      System.exit(0);
   }
  }
 }
}
```

OUTPUT



Current Book Listings

ID	Title	Author	Price (Rs.)	Actions
1	Clean Code	Robert Martin	799.0	
2	The White Tiger	Aravind Adiga	399.0	

Fill to add a new book

Title	Arice
The Great Gatsby	450.00

Search by title or author...

Showing 1-2 of 2 results.

Result:

The Online Book Store web app was built using Hibernate with n-tier architecture. It allows adding, updating, deleting, viewing, and searching books efficiently with ORM integration for smooth database operations.