**Week-4 Exercises**

**Exercise 1: Online Bookstore - Setting Up RESTful Services**

package com.example.bookstoreapi;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class BookstoreApiApplication {

public static void main(String[] args) {

SpringApplication.run(BookstoreApiApplication.class, args);

}

}

**Exercise 2: Online Bookstore - Creating Basic REST Controllers**

package com.example.bookstoreapi.model;

import lombok.Data;

@Data

public class Book {

private Long id;

private String title;

private String author;

private double price;

private String isbn;

}

package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.model.Book;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import java.util.ArrayList;

import java.util.List;

@RestController

@RequestMapping("/books")

public class BookController {

private List<Book> books = new ArrayList<>();

@GetMapping

public List<Book> getAllBooks() {

return books;

}

@PostMapping

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

books.add(book);

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

}

@PutMapping("/{id}")

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

for (Book book : books) {

if (book.getId().equals(id)) {

book.setTitle(updatedBook.getTitle());

book.setAuthor(updatedBook.getAuthor());

book.setPrice(updatedBook.getPrice());

book.setIsbn(updatedBook.getIsbn());

return new ResponseEntity<>(book, HttpStatus.OK);

}

}

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@DeleteMapping("/{id}")

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

books.removeIf(book -> book.getId().equals(id));

return new ResponseEntity<>(HttpStatus.NO\_CONTENT);

}

}

**Exercise 3: Online Bookstore - Handling Path Variables and Query Parameters**

package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.model.Book;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import java.util.ArrayList;

import java.util.List;

@RestController

@RequestMapping("/books")

public class BookController {

private List<Book> books = new ArrayList<>();

@GetMapping("/{id}")

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

for (Book book : books) {

if (book.getId().equals(id)) {

return new ResponseEntity<>(book, HttpStatus.OK);

}

}

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@GetMapping("/search")

public List<Book> searchBooks(@RequestParam(required = false) String title,

@RequestParam(required = false) String author) {

List<Book> result = new ArrayList<>();

for (Book book : books) {

if ((title == null || book.getTitle().equalsIgnoreCase(title)) &&

(author == null || book.getAuthor().equalsIgnoreCase(author))) {

result.add(book);

}

}

return result;

}

}

**Exercise 4: Online Bookstore - Processing Request Body and Form Data**

package com.example.bookstoreapi.model;

import lombok.Data;

@Data

public class Customer {

private Long id;

private String name;

private String email;

}

package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.model.Customer;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import java.util.ArrayList;

import java.util.List;

@RestController

@RequestMapping("/customers")

public class CustomerController {

private List<Customer> customers = new ArrayList<>();

@PostMapping

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

customers.add(customer);

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

}

@PostMapping("/form")

public ResponseEntity<Customer> registerCustomer(@RequestParam String name, @RequestParam String email) {

Customer customer = new Customer();

customer.setName(name);

customer.setEmail(email);

customers.add(customer);

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

}

}

**Exercise 5: Online Bookstore - Customizing Response Status and Headers**

package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.model.Book;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

import java.util.ArrayList;

import java.util.List;

@RestController

@RequestMapping("/books")

public class BookController {

private List<Book> books = new ArrayList<>();

@PostMapping

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

books.add(book);

HttpHeaders headers = new HttpHeaders();

headers.add("Custom-Header", "Book created successfully");

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

}

@PutMapping("/{id}")

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

for (Book book : books) {

if (book.getId().equals(id)) {

book.setTitle(updatedBook.getTitle());

book.setAuthor(updatedBook.getAuthor());

book.setPrice(updatedBook.getPrice());

book.setIsbn(updatedBook.getIsbn());

return new ResponseEntity<>(book, HttpStatus.OK);

}

}

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

@DeleteMapping("/{id}")

@ResponseStatus(HttpStatus.NO\_CONTENT)

public void deleteBook(@PathVariable Long id) {

books.removeIf(book -> book.getId().equals(id));

}

}

**Exercise 6: Online Bookstore - Exception Handling in REST Controllers**

package com.example.bookstoreapi.exception;

import org.springframework.http.HttpStatus;

import org.springframework.web.bind.annotation.ControllerAdvice;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.bind.annotation.ResponseStatus;

@ControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler(BookNotFoundException.class)

@ResponseStatus(HttpStatus.NOT\_FOUND)

public String handleBookNotFoundException(BookNotFoundException ex) {

return ex.getMessage();

}

@ExceptionHandler(Exception.class)

@ResponseStatus(HttpStatus.INTERNAL\_SERVER\_ERROR)

public String handleGeneralException(Exception ex) {

return "An error occurred: " + ex.getMessage();

}

}

package com.example.bookstoreapi.exception;

public class BookNotFoundException extends RuntimeException {

public BookNotFoundException(String message) {

super(message);

}

}

**Exercise 7: Online Bookstore - Introduction to Data Transfer Objects (DTOs)**

package com.example.bookstoreapi.dto;

import lombok.Data;

@Data

public class BookDTO {

private Long id;

private String title;

private String author;

private double price;

}

@Data

public class CustomerDTO {

private Long id;

private String name;

private String email;

}

package com.example.bookstoreapi.mapper;

import com.example.bookstoreapi.dto.BookDTO;

import com.example.bookstoreapi.dto.CustomerDTO;

import com.example.bookstoreapi.model.Book;

import com.example.bookstoreapi.model.Customer;

import org.mapstruct.Mapper;

@Mapper(componentModel = "spring")

public interface BookstoreMapper {

BookDTO toBookDTO(Book book);

Book toBook(BookDTO bookDTO);

CustomerDTO toCustomerDTO(Customer customer);

Customer toCustomer(CustomerDTO customerDTO);

}

package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.dto.BookDTO;

import com.example.bookstoreapi.dto.CustomerDTO;

import com.example.bookstoreapi.mapper.BookstoreMapper;

import com.example.bookstoreapi.model.Book;

import com.example.bookstoreapi.model.Customer;

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.ArrayList;

import java.util.List;

@RestController

@RequestMapping("/books")

public class BookDTOController {

private List<Book> books = new ArrayList<>();

@Autowired

private BookstoreMapper mapper;

@PostMapping

public ResponseEntity<BookDTO> createBook(@RequestBody BookDTO bookDTO) {

Book book = mapper.toBook(bookDTO);

books.add(book);

return new ResponseEntity<>(mapper.toBookDTO(book), HttpStatus.CREATED);

}

@GetMapping("/{id}")

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

for (Book book : books) {

if (book.getId().equals(id)) {

return new ResponseEntity<>(mapper.toBookDTO(book), HttpStatus.OK);

}

}

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

}

@RestController

@RequestMapping("/customers")

public class CustomerDTOController {

private List<Customer> customers = new ArrayList<>();

@Autowired

private BookstoreMapper mapper;

@PostMapping

public ResponseEntity<CustomerDTO> createCustomer(@RequestBody CustomerDTO customerDTO) {

Customer customer = mapper.toCustomer(customerDTO);

customers.add(customer);

return new ResponseEntity<>(mapper.toCustomerDTO(customer), HttpStatus.CREATED);

}

@GetMapping("/{id}")

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

for (Customer customer : customers) {

if (customer.getId().equals(id)) {

return new ResponseEntity<>(mapper.toCustomerDTO(customer), HttpStatus.OK);

}

}

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

}