# Q3.

1. @GetMapping("/{id}")

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

Optional<Book> book = bookList.stream().filter(b -> b.getId().equals(id)).findFirst();

return book.map(ResponseEntity::ok).orElseGet(() -> ResponseEntity.status(HttpStatus.NOT\_FOUND).build());

}

2. @GetMapping("/search")

public ResponseEntity<List<Book>> searchBooks(

@RequestParam(required = false) String title,

@RequestParam(required = false) String author) {

List<Book> filteredBooks = bookList;

if (title != null) {

filteredBooks = filteredBooks.stream()

.filter(book -> book.getTitle().equalsIgnoreCase(title))

.toList();

}

if (author != null) {

filteredBooks = filteredBooks.stream()

.filter(book -> book.getAuthor().equalsIgnoreCase(author))

.toList();

}

return ResponseEntity.ok(filteredBooks);

}

3. 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;

import java.util.Optional;

@RestController

@RequestMapping("/books")

public class BookController {

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

// Constructor with some sample books

public BookController() {

bookList.add(new Book(1L, "The Great Gatsby", "F. Scott Fitzgerald", 10.99, "9780743273565"));

bookList.add(new Book(2L, "1984", "George Orwell", 8.99, "9780451524935"));

}

// GET all books

@GetMapping

public ResponseEntity<List<Book>> getAllBooks() {

return ResponseEntity.ok(bookList);

}

// GET a book by ID

@GetMapping("/{id}")

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

Optional<Book> book = bookList.stream().filter(b -> b.getId().equals(id)).findFirst();

return book.map(ResponseEntity::ok).orElseGet(() -> ResponseEntity.status(HttpStatus.NOT\_FOUND).build());

}

// POST a new book

@PostMapping

public ResponseEntity<Book> addBook(@RequestBody Book newBook) {

bookList.add(newBook);

return ResponseEntity.status(HttpStatus.CREATED).body(newBook);

}

// PUT to update an existing book

@PutMapping("/{id}")

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

Optional<Book> existingBook = bookList.stream().filter(b -> b.getId().equals(id)).findFirst();

if (existingBook.isPresent()) {

bookList.remove(existingBook.get());

bookList.add(updatedBook);

return ResponseEntity.ok(updatedBook);

} else {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).build();

}

}

// DELETE a book by ID

@DeleteMapping("/{id}")

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

boolean removed = bookList.removeIf(book -> book.getId().equals(id));

if (removed) {

return ResponseEntity.noContent().build();

} else {

return ResponseEntity.status(HttpStatus.NOT\_FOUND).build();

}

}

// SEARCH books by title and/or author

@GetMapping("/search")

public ResponseEntity<List<Book>> searchBooks(

@RequestParam(required = false) String title,

@RequestParam(required = false) String author) {

List<Book> filteredBooks = bookList;

if (title != null) {

filteredBooks = filteredBooks.stream()

.filter(book -> book.getTitle().equalsIgnoreCase(title))

.toList();

}

if (author != null) {

filteredBooks = filteredBooks.stream()

.filter(book -> book.getAuthor().equalsIgnoreCase(author))

.toList();

}

return ResponseEntity.ok(filteredBooks);

}

}

# Q4.

1. package com.example.bookstoreapi.model;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Customer {

private Long id;

private String name;

private String email;

private String address;

}

2. 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> customerList = new ArrayList<>();

// POST endpoint to create a new customer using JSON request body

@PostMapping

public ResponseEntity<Customer> addCustomer(@RequestBody Customer newCustomer) {

customerList.add(newCustomer);

return ResponseEntity.status(HttpStatus.CREATED).body(newCustomer);

}

}

3. 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> customerList = new ArrayList<>();

// POST endpoint to create a new customer using JSON request body

@PostMapping

public ResponseEntity<Customer> addCustomer(@RequestBody Customer newCustomer) {

customerList.add(newCustomer);

return ResponseEntity.status(HttpStatus.CREATED).body(newCustomer);

}

// POST endpoint to process form data for customer registrations

@PostMapping("/register")

public ResponseEntity<Customer> registerCustomer(

@RequestParam Long id,

@RequestParam String name,

@RequestParam String email,

@RequestParam String address) {

Customer newCustomer = new Customer(id, name, email, address);

customerList.add(newCustomer);

return ResponseEntity.status(HttpStatus.CREATED).body(newCustomer);

}

}

# Q4.

1. package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.model.Book;

import org.springframework.http.HttpStatus;

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

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/books")

public class BookController {

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

public BookController() {

bookList.add(new Book(1L, "The Great Gatsby", "F. Scott Fitzgerald", 10.99, "9780743273565"));

bookList.add(new Book(2L, "1984", "George Orwell", 8.99, "9780451524935"));

}

// GET a book by ID with custom response status

@GetMapping("/{id}")

@ResponseStatus(HttpStatus.OK)

public Book getBookById(@PathVariable Long id) {

return bookList.stream()

.filter(book -> book.getId().equals(id))

.findFirst()

.orElseThrow(() -> new BookNotFoundException("Book not found with id " + id));

}

@ResponseStatus(HttpStatus.NOT\_FOUND)

@ExceptionHandler(BookNotFoundException.class)

public String handleBookNotFound(BookNotFoundException ex) {

return ex.getMessage();

}

}

package com.example.bookstoreapi.exception;

public class BookNotFoundException extends RuntimeException {

public BookNotFoundException(String message) {

super(message);

}

}

2. 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> bookList = new ArrayList<>();

public BookController() {

bookList.add(new Book(1L, "The Great Gatsby", "F. Scott Fitzgerald", 10.99, "9780743273565"));

bookList.add(new Book(2L, "1984", "George Orwell", 8.99, "9780451524935"));

}

// POST a new book with a custom header

@PostMapping

public ResponseEntity<Book> addBook(@RequestBody Book newBook) {

bookList.add(newBook);

HttpHeaders headers = new HttpHeaders();

headers.add("X-Book-Creation", "Book created successfully");

return ResponseEntity.status(HttpStatus.CREATED)

.headers(headers)

.body(newBook);

}

}

3. package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.exception.BookNotFoundException;

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> bookList = new ArrayList<>();

public BookController() {

bookList.add(new Book(1L, "The Great Gatsby", "F. Scott Fitzgerald", 10.99, "9780743273565"));

bookList.add(new Book(2L, "1984", "George Orwell", 8.99, "9780451524935"));

}

// GET a book by ID with custom response status

@GetMapping("/{id}")

@ResponseStatus(HttpStatus.OK)

public Book getBookById(@PathVariable Long id) {

return bookList.stream()

.filter(book -> book.getId().equals(id))

.findFirst()

.orElseThrow(() -> new BookNotFoundException("Book not found with id " + id));

}

@ResponseStatus(HttpStatus.NOT\_FOUND)

@ExceptionHandler(BookNotFoundException.class)

public String handleBookNotFound(BookNotFoundException ex) {

return ex.getMessage();

}

// POST a new book with a custom header

@PostMapping

public ResponseEntity<Book> addBook(@RequestBody Book newBook) {

bookList.add(newBook);

HttpHeaders headers = new HttpHeaders();

headers.add("X-Book-Creation", "Book created successfully");

return ResponseEntity.status(HttpStatus.CREATED)

.headers(headers)

.body(newBook);

}

}

# Q6.

1. package com.example.bookstoreapi.dto;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

public class BookDTO {

private Long id;

private String title;

private String author;

private Double price;

}

package com.example.bookstoreapi.dto;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

public class CustomerDTO {

private Long id;

private String name;

private String email;

}

2. package com.example.bookstoreapi.mapper;

import com.example.bookstoreapi.dto.BookDTO;

import com.example.bookstoreapi.model.Book;

import org.mapstruct.Mapper;

import org.mapstruct.factory.Mappers;

@Mapper

public interface BookMapper {

BookMapper INSTANCE = Mappers.getMapper(BookMapper.class);

BookDTO bookToBookDTO(Book book);

Book bookDTOToBook(BookDTO bookDTO);

}

package com.example.bookstoreapi.mapper;

import com.example.bookstoreapi.dto.CustomerDTO;

import com.example.bookstoreapi.model.Customer;

import org.mapstruct.Mapper;

import org.mapstruct.factory.Mappers;

@Mapper

public interface CustomerMapper {

CustomerMapper INSTANCE = Mappers.getMapper(CustomerMapper.class);

CustomerDTO customerToCustomerDTO(Customer customer);

Customer customerDTOToCustomer(CustomerDTO customerDTO);

}

package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.dto.BookDTO;

import com.example.bookstoreapi.mapper.BookMapper;

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;

import java.util.stream.Collectors;

@RestController

@RequestMapping("/books")

public class BookController {

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

public BookController() {

bookList.add(new Book(1L, "The Great Gatsby", "F. Scott Fitzgerald", 10.99, "9780743273565"));

bookList.add(new Book(2L, "1984", "George Orwell", 8.99, "9780451524935"));

}

// GET all books and return as DTOs

@GetMapping

public List<BookDTO> getAllBooks() {

return bookList.stream()

.map(BookMapper.INSTANCE::bookToBookDTO)

.collect(Collectors.toList());

}

// POST a new book using BookDTO

@PostMapping

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

Book newBook = BookMapper.INSTANCE.bookDTOToBook(bookDTO);

bookList.add(newBook);

return ResponseEntity.status(HttpStatus.CREATED).body(BookMapper.INSTANCE.bookToBookDTO(newBook));

}

}

3. package com.example.bookstoreapi.dto;

import com.fasterxml.jackson.annotation.JsonIgnore;

import com.fasterxml.jackson.annotation.JsonProperty;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

public class BookDTO {

private Long id;

@JsonProperty("book\_title")

private String title;

private String author;

@JsonIgnore

private Double price;

}

package com.example.bookstoreapi.dto;

import com.fasterxml.jackson.annotation.JsonIgnore;

import com.fasterxml.jackson.annotation.JsonProperty;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

public class CustomerDTO {

private Long id;

@JsonProperty("customer\_name")

private String name;

@JsonIgnore

private String email;

}