WEEK 4 EXERCISE

Exercise 1: Online Bookstore - Setting Up RESTful Services Business Scenario

Model

```
package com.example.bookstoreapi.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
public class Book {

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

    private String title;
    private String author;
    private String isbn;
    private double price;
}
```

Repository

```
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> {
}
```

Service

```
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;

@Service
public class BookService {

  @Autowired
  private BookRepository bookRepository;

  public List<Book> getAllBooks() {
    return bookRepository.findAll();
  }

  public Book getBookById(Long id) {
    return bookRepository.findById(id).orElse(null);
  }

  public Book saveBook(Book book) {
    return bookRepository.save(book);
  }

  public void deleteBook(Long id) {
    bookRepository.deleteById(id);
  }
}
```

Controller

```
package com.example.bookstoreapi.controller;
```

```
import com.example.bookstoreapi.model.Book;
import com.example.bookstoreapi.service.BookService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/books")
public class BookController {

    @Autowired
    private BookService bookService;

    @GetMapping
    public List<Book> getAllBooks() {
        return bookService.getAllBooks();
    }

    @GetMapping("/{id}")
    public Book getBookById(@PathVariable Long id) {
        return bookService.getBookById(id);
    }

    @PostMapping
    public Book createBook(@RequestBody Book book) {
        return bookService.saveBook(book);
    }

    @DeleteMapping("/{id}")
    public void deleteBook(@PathVariable Long id) {
        bookService.deleteBook(id);
    }
}
```

Application Class

```
package com.example.bookstoreapi;
```

```java
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 Business Scenario

BookController.java

```
package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.model.Book;
import com.example.bookstoreapi.service.BookService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/books")
public class BookController {

    @Autowired
    private BookService bookService;

    // GET /books - Retrieve all books
    @GetMapping
    public List<Book> getAllBooks() {
        return bookService.getAllBooks();
    }

    // GET /books/{id} - Retrieve a book by ID
    @GetMapping("/{id}")
    public ResponseEntity<Book> getBookById(@PathVariable Long id) {
        Book book = bookService.getBookById(id);
        return book != null ? ResponseEntity.ok(book) : ResponseEntity.notFound().build();
    }

    // POST /books - Create a new book
    @PostMapping
    public ResponseEntity<Book> createBook(@RequestBody Book book) {
        Book savedBook = bookService.saveBook(book);
        return ResponseEntity.ok(savedBook);
    }
```

```
   // PUT /books/{id} - Update an existing book
   @PutMapping("/{id}")
   public ResponseEntity<Book> updateBook(@PathVariable Long id, @RequestBody Book
bookDetails) {
       Book updatedBook = bookService.updateBook(id, bookDetails);
       return updatedBook != null ? ResponseEntity.ok(updatedBook) :
ResponseEntity.notFound().build();
   }
```

Book.java

```
package com.example.bookstoreapi.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
public class Book {

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

   private String title;
   private String author;
   private double price;
   private String isbn;
}
```

BookService

```
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;

@Service
public class BookService {

    @Autowired
    private BookRepository bookRepository;

    public List<Book> getAllBooks() {
        return bookRepository.findAll();
    }

    public Book getBookById(Long id) {
        return bookRepository.findById(id).orElse(null);
    }

    public Book saveBook(Book book) {
        return bookRepository.save(book);
    }

    public Book updateBook(Long id, Book bookDetails) {
        return bookRepository.findById(id).map(book -> {
            book.setTitle(bookDetails.getTitle());
            book.setAuthor(bookDetails.getAuthor());
            book.setPrice(bookDetails.getPrice());
            book.setIsbn(bookDetails.getIsbn());
            return bookRepository.save(book);
        }).orElse(null);
    }

    public boolean deleteBook(Long id) {
        return bookRepository.findById(id).map(book -> {
            bookRepository.delete(book);
            return true;
        }).orElse(false);
    }
}
```

Json

```
{
  "title": "Book Title",
  "author": "Author Name",
  "price": 29.99,
  "isbn": "123-4567890123"
}
{
  "title": "Updated Title",
  "author": "Updated Author",
  "price": 35.99,
  "isbn": "123-4567890123"
}
```

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

BookController.java

```
package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.model.Book;
import com.example.bookstoreapi.service.BookService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/books")
public class BookController {

    @Autowired
    private BookService bookService;

    // GET /books - Retrieve all books
    @GetMapping
    public List<Book> getAllBooks() {
        return bookService.getAllBooks();
    }

    // GET /books/{id} - Retrieve a book by ID using a path variable
    @GetMapping("/{id}")
    public ResponseEntity<Book> getBookById(@PathVariable Long id) {
        Book book = bookService.getBookById(id);
        return book != null ? ResponseEntity.ok(book) : ResponseEntity.notFound().build();
    }

    // GET /books/search - Retrieve books filtered by title and author using query parameters
    @GetMapping("/search")
    public ResponseEntity<List<Book>> searchBooks(
            @RequestParam(required = false) String title,
            @RequestParam(required = false) String author) {
        List<Book> books = bookService.searchBooks(title, author);
        return ResponseEntity.ok(books);
    }

    // POST /books - Create a new book
    @PostMapping
    public ResponseEntity<Book> createBook(@RequestBody Book book) {
        Book savedBook = bookService.saveBook(book);
        return ResponseEntity.ok(savedBook);
    }

    // PUT /books/{id} - Update an existing book
    @PutMapping("/{id}")
    public ResponseEntity<Book> updateBook(@PathVariable Long id, @RequestBody Book
bookDetails) {
        Book updatedBook = bookService.updateBook(id, bookDetails);
        return updatedBook != null ? ResponseEntity.ok(updatedBook) :
ResponseEntity.notFound().build();
    }

    // DELETE /books/{id} - Delete a book by ID
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteBook(@PathVariable Long id) {
        boolean isDeleted = bookService.deleteBook(id);
        return isDeleted ? ResponseEntity.noCo
```

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;

@Service
public class BookService {

    @Autowired
    private BookRepository bookRepository;
```

```
public List<Book> getAllBooks() {
    return bookRepository.findAll();
}

public Book getBookById(Long id) {
    return bookRepository.findById(id).orElse(null);
}

public Book saveBook(Book book) {
    return bookRepository.save(book);
}

public Book updateBook(Long id, Book bookDetails) {
    return bookRepository.findById(id).map(book -> {
        book.setTitle(bookDetails.getTitle());
        book.setAuthor(bookDetails.getAuthor());
        book.setPrice(bookDetails.getPrice());
        book.setIsbn(bookDetails.getIsbn());
        return bookRepository.save(book);
    }).orElse(null);
}

public boolean deleteBook(Long id) {
    return bookRepository.findById(id).map(book -> {
        bookRepository.delete(book);
        return true;
    }).orElse(false);
}

// Search books by title and author
public List<Book> searchBooks(String title, String author) {
    if (title != null && author != null) {
        return bookRepository.findByTitleContainingAndAuthorContaining(title, author);
    } else if (title != null) {
        return bookRepository.findByTitleContaining(title);
    } else if (author != null) {
        return bookRepository.findByAuthorContaining(author);
    } else {
        return bookRepository.findAll();
        }
    }
}
```

BookRepository.java

```
package com.example.bookstoreapi.repository;

import com.example.bookstoreapi.model.Book;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface BookRepository extends JpaRepository<Book, Long> {

    List<Book> findByTitleContaining(String title);
    List<Book> findByAuthorContaining(String author);
    List<Book> findByTitleContainingAndAuthorContaining(String title, String author);
}
```

Json file

```
[
    {
        "id": 1,
        "title": "Spring in Action",
        "author": "Craig Walls",
        "price": 49.99,
        "isbn": "978-1617294945"
    },
    {
        "id": 2,
        "title": "Spring Boot in Practice",
        "author": "Somnath Musib",
        "price": 45.00,
        "isbn": "978-1617298813"
    }
]
\
```

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

Customer.java

```
package com.example.bookstoreapi.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
public class Customer {

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

    private String name;
    private String email;
    private String password;
}
```

CustomerController.java

```
package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.model.Customer;
import com.example.bookstoreapi.service.CustomerService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import java.util.Map;

@RestController
@RequestMapping("/customers")
public class CustomerController {

    @Autowired
    private CustomerService customerService;

    // POST /customers - Create a new customer by accepting a JSON request body
    @PostMapping
    public ResponseEntity<Customer> createCustomer(@RequestBody Customer customer) {
        Customer savedCustomer = customerService.saveCustomer(customer);
        return ResponseEntity.ok(savedCustomer);
    }

    // POST /customers/register - Create a new customer by accepting form data
    @PostMapping("/register")
    public ResponseEntity<Customer> registerCustomer(
            @RequestParam("name") String name,
            @RequestParam("email") String email,
            @RequestParam("password") String password) {
        Customer customer = new Customer();
        customer.setName(name);
        customer.setEmail(email);
        customer.setPassword(password);

        Customer savedCustomer = customerService.saveCustomer(customer);
        return ResponseEntity.ok(savedCustomer);
    }
}
```

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;
```

```
@Service
public class CustomerService {

    @Autowired
    private CustomerRepository customerRepository;

    public Customer saveCustomer(Customer customer) {
        return customerRepository.save(customer);
    }
}
```

CustomerRepository.java

```
package com.example.bookstoreapi.repository;

import com.example.bookstoreapi.model.Customer;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface CustomerRepository extends JpaRepository<Customer, Long> {
}
```

Json

```
{
    "name": "John Doe",
    "email": "john.doe@example.com",
    "password": "password123"
}
{
    "id": 1,
    "name": "John Doe",
    "email": "john.doe@example.com",
    "password": "password123"
}
{
    "id": 2,
    "name": "John Doe",
    "email": "john.doe@example.com",
    "password": "password123"}
```

Exercise 5: Online Bookstore - Customizing Response Status and Headers
Business Scenario:

BookController.java

```
package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.model.Book;
import com.example.bookstoreapi.service.BookService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/books")
public class BookController {

    @Autowired
    private BookService bookService;

    // GET /books - Retrieve all books
    @GetMapping
    public ResponseEntity<List<Book>> getAllBooks() {
        List<Book> books = bookService.getAllBooks();
        HttpHeaders headers = new HttpHeaders();
        headers.add("Custom-Header", "BookListHeader");
        return new ResponseEntity<>(books, headers, HttpStatus.OK);
    }

    // GET /books/{id} - Retrieve a book by ID using a path variable
    @GetMapping("/{id}")
    @ResponseStatus(HttpStatus.OK)
    public ResponseEntity<Book> getBookById(@PathVariable Long id) {
        Book book = bookService.getBookById(id);
        if (book != null) {
            HttpHeaders headers = new HttpHeaders();
            headers.add("Custom-Header", "BookDetailsHeader");
            return new ResponseEntity<>(book, headers, HttpStatus.OK);
        } else {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }

    // POST /books - Create a new book
    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    public ResponseEntity<Book> createBook(@RequestBody Book book) {
        Book savedBook = bookService.saveBook(book);
        HttpHeaders headers = new HttpHeaders();
        headers.add("Custom-Header", "BookCreatedHeader");
        return new ResponseEntity<>(savedBook, headers, HttpStatus.CREATED);
    }

    // PUT /books/{id} - Update an existing book
    @PutMapping("/{id}")
    public ResponseEntity<Book> updateBook(@PathVariable Long id, @RequestBody Book
bookDetails) {
        Book updatedBook = bookService.updateBook(id, bookDetails);
        if (updatedBook != null) {
            HttpHeaders headers = new HttpHeaders();
            headers.add("Custom-Header", "BookUpdatedHeader");
            return new ResponseEntity<>(updatedBook, headers, HttpStatus.OK);
        } else {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }

    // DELETE /books/{id} - Delete a book by ID
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteBook(@PathVariable Long id) {
        boolean isDeleted = bookService.deleteBook(id);
        if (isDeleted) {
            HttpHeaders headers = new HttpHeaders();
            headers.add("Custom-Header", "BookDeletedHeader");
            return new ResponseEntity<>(headers, HttpStatus.NO_CONTENT);
        } else {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }
}
```

Json

```
{
    "id": 1,
    "title": "Spring in Action",
    "author": "Craig Walls",
    "price": 49.99,
    "isbn": "978-1617294945"
}
{
    "id": 2,
    "title": "Spring Boot in Practice",
    "author": "Somnath Musib",
    "price": 45.00,
    "isbn": "978-1617298813"
}
```

Exercise 6: Online Bookstore - Exception Handling in REST Controllers Business Scenario:

GlobalExceptionHandler.java

```
package com.example.bookstoreapi.exception;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.FieldError;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.context.request.WebRequest;

import javax.persistence.EntityNotFoundException;
import java.util.HashMap;
import java.util.Map;

@ControllerAdvice
public class GlobalExceptionHandler {

   // Handle EntityNotFoundException
   @ExceptionHandler(EntityNotFoundException.class)
   public ResponseEntity<String> handleEntityNotFoundException(EntityNotFoundException
ex, WebRequest request) {
       return new ResponseEntity<>("Resource not found: " + ex.getMessage(),
HttpStatus.NOT_FOUND);
   }

   // Handle MethodArgumentNotValidException
   @ExceptionHandler(MethodArgumentNotValidException.class)
   public ResponseEntity<Map<String, String>>
handleValidationExceptions(MethodArgumentNotValidException ex) {
       Map<String, String> errors = new HashMap<>();
       ex.getBindingResult().getAllErrors().forEach((error) -> {
          String fieldName = ((FieldError) error).getField();
          String errorMessage = error.getDefaultMessage();
          errors.put(fieldName, errorMessage);
       });
       return new ResponseEntity<>(errors, HttpStatus.BAD_REQUEST);
```

```
    }

    // Handle other exceptions
    @ExceptionHandler(Exception.class)
    public ResponseEntity<String> handleGlobalException(Exception ex, WebRequest request) {
        return new ResponseEntity<>("An error occurred: " + ex.getMessage(),
HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

Modified BookController.java

```
package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.model.Book;
import com.example.bookstoreapi.service.BookService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import javax.persistence.EntityNotFoundException;
import java.util.List;

@RestController
@RequestMapping("/books")
public class BookController {

    @Autowired
    private BookService bookService;

    // GET /books - Retrieve all books
    @GetMapping
    public ResponseEntity<List<Book>> getAllBooks() {
        List<Book> books = bookService.getAllBooks();
        HttpHeaders headers = new HttpHeaders();
        headers.add("Custom-Header", "BookListHeader");
        return new ResponseEntity<>(books, headers, HttpStatus.OK);
    }
```

```
  // GET /books/{id} - Retrieve a book by ID using a path variable
  @GetMapping("/{id}")
  public ResponseEntity<Book> getBookById(@PathVariable Long id) {
    Book book = bookService.getBookById(id);
    if (book == null) {
      throw new EntityNotFoundException("Book with ID " + id + " not found");
    }
    HttpHeaders headers = new HttpHeaders();
    headers.add("Custom-Header", "BookDetailsHeader");
    return new ResponseEntity<>(book, headers, HttpStatus.OK);
  }

  // POST /books - Create a new book
  @PostMapping
  public ResponseEntity<Book> createBook(@RequestBody Book book) {
    Book savedBook = bookService.saveBook(book);
    HttpHeaders headers = new HttpHeaders();
    headers.add("Custom-Header", "BookCreatedHeader");
    return new ResponseEntity<>(savedBook, headers, HttpStatus.CREATED);
  }

  // PUT /books/{id} - Update an existing book
  @PutMapping("/{id}")
  public ResponseEntity<Book> updateBook(@PathVariable Long id, @RequestBody Book
bookDetails) {
    Book updatedBook = bookService.updateBook(id, bookDetails);
    if (updatedBook == null) {
      throw new EntityNotFoundException("Book with ID " + id + " not found");
    }
    HttpHeaders headers = new HttpHeaders();
    headers.add("Custom-Header", "BookUpdatedHeader");
    return new ResponseEntity<>(updatedBook, headers, HttpStatus.OK);
  }

  // DELETE /books/{id} - Delete a book by ID
  @DeleteMapping("/{id}")
  public ResponseEntity<Void> deleteBook(@PathVariable Long id) {
    boolean isDeleted = bookService.deleteBook(id);
    if (!isDeleted) {
      throw new EntityNotFoundException("Book with ID " + id + " not found");
```

```
        }
        HttpHeaders headers = new HttpHeaders();
        headers.add("Custom-Header", "BookDeletedHeader");
        return new ResponseEntity<>(headers, HttpStatus.NO_CONTENT);
    }
}
```

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

BookDTO.java

```
package com.example.bookstoreapi.dto;

public class BookDTO {

    private Long id;
    private String title;
    private String author;
    private Double price;
    private String isbn;

    // 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 String getIsbn() {
        return isbn;
    }

    public void setIsbn(String isbn) {
        this.isbn = isbn;
    }
}
```

CustomerDTO.java

```
package com.example.bookstoreapi.dto;

public class CustomerDTO {

    private Long id;
    private String name;
    private String email;
    private String phoneNumber;

    // 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 String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }
}
```

Pom.xml

```
<dependency>
    <groupId>org.mapstruct</groupId>
    <artifactId>mapstruct</artifactId>
    <version>1.5.3.Final</version>
</dependency>
<dependency>
    <groupId>org.mapstruct</groupId>
    <artifactId>mapstruct-processor</artifactId>
    <version>1.5.3.Final</version>
    <scope>provided</scope>
</dependency>
```

BookMapper.java

```
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);
}
```

CustomerMapper.java

```
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);
}
```

Customized customerDTO.java

```
package com.example.bookstoreapi.dto;

import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonProperty;
```

```
public class CustomerDTO {

    private Long id;

    @JsonProperty("full_name")
    private String name;

    @JsonProperty("email_address")
    private String email;

    @JsonIgnore
    private String phoneNumber;

    // 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 String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }
}
```

Modified BookController.java

```
package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.dto.BookDTO;
import com.example.bookstoreapi.mapper.BookMapper;
import com.example.bookstoreapi.model.Book;
import com.example.bookstoreapi.service.BookService;
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.stream.Collectors;

@RestController
@RequestMapping("/books")
public class BookController {

    @Autowired
    private BookService bookService;

    private final BookMapper bookMapper = BookMapper.INSTANCE;

    // GET /books - Retrieve all books
    @GetMapping
    public ResponseEntity<List<BookDTO>> getAllBooks() {
        List<Book> books = bookService.getAllBooks();
        List<BookDTO> bookDTOs = books.stream()
                                .map(bookMapper::bookToBookDTO)
                                .collect(Collectors.toList());
        return new ResponseEntity<>(bookDTOs, HttpStatus.OK);
    }
    // GET /books/{id} - Retrieve a book by ID
    @GetMapping("/{id}")
    public ResponseEntity<BookDTO> getBookById(@PathVariable Long id) {
        Book book = bookService.getBookById(id);
        if (book == null) {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
        BookDTO bookDTO = bookMapper.bookToBookDTO(book);
        return new ResponseEntity<>(bookDTO, HttpStatus.OK);
    }

    // POST /books - Create a new book
    @PostMapping
    public ResponseEntity<BookDTO> createBook(@RequestBody BookDTO bookDTO) {
        Book book = bookMapper.bookDTOToBook(bookDTO);
        Book savedBook = bookService.saveBook(book);
        BookDTO savedBookDTO = bookMapper.bookToBookDTO(savedBook);
        return new ResponseEntity<>(savedBookDTO, HttpStatus.CREATED);
    }

    // PUT /books/{id} - Update an existing book
    @PutMapping("/{id}")
    public ResponseEntity<BookDTO> updateBook(@PathVariable Long id, @RequestBody
BookDTO bookDTO) {
        Book book = bookMapper.bookDTOToBook(bookDTO);
        Book updatedBook = bookService.updateBook(id, book);
        if (updatedBook == null) {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
        BookDTO updatedBookDTO = bookMapper.bookToBookDTO(updatedBook);
        return new ResponseEntity<>(updatedBookDTO, HttpStatus.OK);
    }

    // DELETE /books/{id} - Delete a book by ID
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteBook(@PathVariable Long id) {
        boolean isDeleted = bookService.deleteBook(id);
        if (!isDeleted) {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
    }
}
```