<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.dataformat</groupId>

<artifactId>jackson-dataformat-xml</artifactId>

</dependency>

application.properties

spring.mvc.contentnegotiation.favor-path-extension=true

spring.mvc.contentnegotiation.favor-parameter=false

spring.mvc.contentnegotiation.parameter-name=mediaType

spring.mvc.contentnegotiation.ignore-unknown-path-extensions=true

spring.mvc.contentnegotiation.media-types.json=application/json

spring.mvc.contentnegotiation.media-types.xml=application/xml

application.yml

spring:

mvc:

contentnegotiation:

favor-path-extension: true

favor-parameter: false

parameter-name: mediaType

ignore-unknown-path-extensions: true

media-types:

json: application/json

xml: application/xml

Update BookController

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.resource.BookResource;

import com.example.bookstoreapi.service.BookService;

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

import org.springframework.hateoas.CollectionModel;

import org.springframework.hateoas.EntityModel;

import org.springframework.http.MediaType;

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

import javax.validation.Valid;

import java.util.List;

import java.util.stream.Collectors;

@RestController

@RequestMapping("/books")

public class BookController {

@Autowired

private BookService bookService;

@Autowired

private BookMapper bookMapper;

@PostMapping(consumes = {MediaType.APPLICATION\_JSON\_VALUE, MediaType.APPLICATION\_XML\_VALUE},

produces = {MediaType.APPLICATION\_JSON\_VALUE, MediaType.APPLICATION\_XML\_VALUE})

public EntityModel<BookResource> addBook(@Valid @RequestBody BookDTO bookDTO) {

Book book = bookMapper.toBook(bookDTO);

Book savedBook = bookService.addBook(book);

BookDTO savedBookDTO = bookMapper.toBookDTO(savedBook);

BookResource bookResource = new BookResource(savedBookDTO);

bookResource.addLinks();

return EntityModel.of(bookResource);

}

@GetMapping(value = "/{id}", produces = {MediaType.APPLICATION\_JSON\_VALUE, MediaType.APPLICATION\_XML\_VALUE})

public EntityModel<BookResource> getBookById(@PathVariable Long id) {

Book book = bookService.getBookById(id)

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

BookDTO bookDTO = bookMapper.toBookDTO(book);

BookResource bookResource = new BookResource(bookDTO);

bookResource.addLinks();

return EntityModel.of(bookResource);

}

@GetMapping(produces = {MediaType.APPLICATION\_JSON\_VALUE, MediaType.APPLICATION\_XML\_VALUE})

public CollectionModel<BookResource> getAllBooks() {

List<BookDTO> books = bookService.getAllBooks().stream()

.map(bookMapper::toBookDTO)

.collect(Collectors.toList());

List<BookResource> bookResources = books.stream()

.map(BookResource::new)

.peek(BookResource::addLinks)

.collect(Collectors.toList());

return CollectionModel.of(bookResources);

}

@PutMapping(value = "/{id}", consumes = {MediaType.APPLICATION\_JSON\_VALUE, MediaType.APPLICATION\_XML\_VALUE},

produces = {MediaType.APPLICATION\_JSON\_VALUE, MediaType.APPLICATION\_XML\_VALUE})

public EntityModel<BookResource> updateBook(@PathVariable Long id, @Valid @RequestBody BookDTO bookDTO) {

Book updatedBook = bookService.updateBook(id, bookMapper.toBook(bookDTO));

BookDTO updatedBookDTO = bookMapper.toBookDTO(updatedBook);

BookResource bookResource = new BookResource(updatedBookDTO);

bookResource.addLinks();

return EntityModel.of(bookResource);

}

@DeleteMapping("/{id}")

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

bookService.deleteBook(id);

return ResponseEntity.noContent().build();

}

}

package com.example.bookstoreapi.controller;

import com.example.bookstoreapi.dto.CustomerDTO;

import com.example.bookstoreapi.mapper.CustomerMapper;

import com.example.bookstoreapi.model.Customer;

import com.example.bookstoreapi.resource.CustomerResource;

import com.example.bookstoreapi.service.CustomerService;

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

import org.springframework.hateoas.CollectionModel;

import org.springframework.hateoas.EntityModel;

import org.springframework.http.MediaType;

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

import javax.validation.Valid;

import java.util.List;

import java.util.stream.Collectors;

@RestController

@RequestMapping("/customers")

public class CustomerController {

@Autowired

private CustomerService customerService;

@Autowired

private CustomerMapper customerMapper;

@PostMapping(consumes = {MediaType.APPLICATION\_JSON\_VALUE, MediaType.APPLICATION\_XML\_VALUE},

produces = {MediaType.APPLICATION\_JSON\_VALUE, MediaType.APPLICATION\_XML\_VALUE})

public EntityModel<CustomerResource> addCustomer(@Valid @RequestBody CustomerDTO customerDTO) {

Customer customer = customerMapper.toCustomer(customerDTO);

Customer savedCustomer = customerService.addCustomer(customer);

CustomerDTO savedCustomerDTO = customerMapper.toCustomerDTO(savedCustomer);

CustomerResource customerResource = new CustomerResource(savedCustomerDTO);

customerResource.addLinks();

return EntityModel.of(customerResource);

}

@GetMapping(value = "/{id}", produces = {MediaType.APPLICATION\_JSON\_VALUE, MediaType.APPLICATION\_XML\_VALUE})

public EntityModel<CustomerResource> getCustomerById(@PathVariable Long id) {

Customer customer = customerService.getCustomerById(id)

.orElseThrow(() -> new RuntimeException("Customer not found with id " + id));

CustomerDTO customerDTO = customerMapper.toCustomerDTO(customer);

CustomerResource customerResource = new CustomerResource(customerDTO);

customerResource.addLinks();

return EntityModel.of(customerResource);

}

@GetMapping(produces = {MediaType.APPLICATION\_JSON\_VALUE, MediaType.APPLICATION\_XML\_VALUE})

public CollectionModel<CustomerResource> getAllCustomers() {

List<CustomerDTO> customers = customerService.getAllCustomers().stream()

.map(customerMapper::toCustomerDTO)

.collect(Collectors.toList());

List<CustomerResource> customerResources = customers.stream()

.map(CustomerResource::new)

.peek(CustomerResource::addLinks)

.collect(Collectors.toList());

return CollectionModel.of(customerResources);

}

@PutMapping(value = "/{id}", consumes = {MediaType.APPLICATION\_JSON\_VALUE, MediaType.APPLICATION\_XML\_VALUE},

produces = {MediaType.APPLICATION\_JSON\_VALUE, MediaType.APPLICATION\_XML\_VALUE})

public EntityModel<CustomerResource> updateCustomer(@PathVariable Long id, @Valid @RequestBody CustomerDTO customerDTO) {

Customer updatedCustomer = customerService.updateCustomer(id, customerMapper.toCustomer(customerDTO));

CustomerDTO updatedCustomerDTO = customerMapper.toCustomerDTO(updatedCustomer);

CustomerResource customerResource = new CustomerResource(updatedCustomerDTO);

customerResource.addLinks();

return EntityModel.of(customerResource);

}

@DeleteMapping("/{id}")

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

customerService.deleteCustomer(id);

return ResponseEntity.noContent().build();

}

}