<dependency>

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

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

</dependency>

package com.example.bookstoreapi.resource;

import com.example.bookstoreapi.dto.BookDTO;

import org.springframework.hateoas.RepresentationModel;

import org.springframework.hateoas.server.mvc.WebMvcLinkBuilder;

public class BookResource extends RepresentationModel<BookResource> {

private final BookDTO bookDTO;

public BookResource(BookDTO bookDTO) {

this.bookDTO = bookDTO;

}

public BookDTO getBookDTO() {

return bookDTO;

}

public void addLinks() {

// Add link to self

this.add(WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(BookController.class).getBookById(bookDTO.getId())).withSelfRel());

// Add link to update

this.add(WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(BookController.class).updateBook(bookDTO.getId(), bookDTO)).withRel("update"));

// Add link to delete

this.add(WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(BookController.class).deleteBook(bookDTO.getId())).withRel("delete"));

}

}

package com.example.bookstoreapi.resource;

import com.example.bookstoreapi.dto.CustomerDTO;

import org.springframework.hateoas.RepresentationModel;

import org.springframework.hateoas.server.mvc.WebMvcLinkBuilder;

public class CustomerResource extends RepresentationModel<CustomerResource> {

private final CustomerDTO customerDTO;

public CustomerResource(CustomerDTO customerDTO) {

this.customerDTO = customerDTO;

}

public CustomerDTO getCustomerDTO() {

return customerDTO;

}

public void addLinks() {

// Add link to self

this.add(WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(CustomerController.class).getCustomerById(customerDTO.getId())).withSelfRel());

// Add link to update

this.add(WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(CustomerController.class).updateCustomer(customerDTO.getId(), customerDTO)).withRel("update"));

// Add link to delete

this.add(WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(CustomerController.class).deleteCustomer(customerDTO.getId())).withRel("delete"));

}

}

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.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

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("/{id}")

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

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("/{id}")

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.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

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("/{id}")

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

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("/{id}")

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();

}

}