**Exercise 9: Online Bookstore - Understanding HATEOAS**

1. **Add Links to Resources Using Spring HATEOAS**

Add Spring HATEOAS Dependency

<dependency>

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

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

</dependency>

1. **Enhance the Book and Customer Resources with Links**

Book Resource with HATEOAS

import org.springframework.hateoas.EntityModel;

import org.springframework.hateoas.Link;

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

@RestController

@RequestMapping("/api/books")

public class BookController {

@Autowired

private BookService bookService;

// Get a book by ID with HATEOAS links

@GetMapping("/{id}")

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

Book book = bookService.getBookById(id);

// Create an EntityModel for the book and add links

EntityModel<Book> bookModel = EntityModel.of(book);

Link selfLink = WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(BookController.class).getBookById(id)).withSelfRel();

Link allBooksLink = WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(BookController.class).getAllBooks()).withRel("all-books");

bookModel.add(selfLink, allBooksLink);

return ResponseEntity.ok(bookModel);

}

// Get all books

@GetMapping

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

List<Book> books = bookService.getAllBooks();

return ResponseEntity.ok(books);

}

}

Customer Resource with HATEOAS

@RestController

@RequestMapping("/api/customers")

public class CustomerController {

@Autowired

private CustomerService customerService;

// Get a customer by ID with HATEOAS links

@GetMapping("/{id}")

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

Customer customer = customerService.getCustomerById(id);

// Create an EntityModel for the customer and add links

EntityModel<Customer> customerModel = EntityModel.of(customer);

Link selfLink = WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(CustomerController.class).getCustomerById(id)).withSelfRel();

Link allCustomersLink = WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(CustomerController.class).getAllCustomers()).withRel("all-customers");

customerModel.add(selfLink, allCustomersLink);

return ResponseEntity.ok(customerModel);

}

// Get all customers

@GetMapping

public ResponseEntity<List<Customer>> getAllCustomers() {

List<Customer> customers = customerService.getAllCustomers();

return ResponseEntity.ok(customers);

}

}

1. **Hypermedia-Driven APIs**

{

"id": 1,

"title": "Effective Java",

"author": "Joshua Bloch",

"price": 45.99,

"\_links": {

"self": {

"href": "http://localhost:8080/api/books/1"

},

"all-books": {

"href": "http://localhost:8080/api/books"

}

}

}

1. **Building Hypermedia-Driven APIs**
2. Service Layer Update