Add links to resources using spring HATEOAS

<dependency>

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

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

</dependency>

Enhance DTO with HATEOAS Links

import org.springframework.hateoas.RepresentationModel;

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

public class BookDTO extends RepresentationModel<BookDTO> {

private Long id;

private String title;

private String author;

private int price;

// Getters and setters

}

public class CustomerDTO extends RepresentationModel<CustomerDTO> {

private Long id;

private String name;

private String email;

// Getters and setters

}

Add links to controller

@RestController

@RequestMapping("/api/books")

public class BookController {

@Autowired

private BookService bookService;

@GetMapping("/{id}")

public ResponseEntity<BookDTO> getBook(@PathVariable Long id) {

BookDTO bookDTO = bookService.getBook(id);

// Add self link

bookDTO.add(WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(BookController.class).getBook(id)).withSelfRel());

// Add other relevant links

bookDTO.add(WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(BookController.class).getAllBooks()).withRel("all-books"));

return new ResponseEntity<>(bookDTO, HttpStatus.OK);

}

@GetMapping

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

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

books.forEach(book -> {

book.add(WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(BookController.class).getBook(book.getId())).withSelfRel());

});

return new ResponseEntity<>(books, HttpStatus.OK);

}

}

@RestController

@RequestMapping("/api/customers")

public class CustomerController {

@Autowired

private CustomerService customerService;

@GetMapping("/{id}")

public ResponseEntity<CustomerDTO> getCustomer(@PathVariable Long id) {

CustomerDTO customerDTO = customerService.getCustomer(id);

// Add self link

customerDTO.add(WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(CustomerController.class).getCustomer(id)).withSelfRel());

// Add other relevant links

customerDTO.add(WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(CustomerController.class).getAllCustomers()).withRel("all-customers"));

return new ResponseEntity<>(customerDTO, HttpStatus.OK);

}

@GetMapping

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

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

customers.forEach(customer -> {

customer.add(WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(CustomerController.class).getCustomer(customer.getId())).withSelfRel());

});

return new ResponseEntity<>(customers, HttpStatus.OK); } }

Building and Consuming Hypermedia driven APIs

{

"id": 1,

"title": "Spring in Action",

"author": "Craig Walls",

"price": 50,

"\_links": {

"self": {

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

},

"all-books": {

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

}

}

}

Consume Hypermedia-Driven APIs

fetch('http://localhost:8080/api/books/1')

.then(response => response.json())

.then(book => {

console.log(book.title); // Use the book data

return fetch(book.\_links['all-books'].href); // Navigate to all books using the link

})

.then(response => response.json())

.then(allBooks => console.log(allBooks));