## Міністерство освіти і науки України

# Національний університет “Львівська політехніка”

Кафедра інформаційних систем та мереж



Прикладне програмування

Звіт

До лабораторної роботи № 4

«Maven та Spring»

Виконав:

ст. гр. ІТ–22

Петровський Н.

Прийняла:

Завущак І. І.

«\_\_\_» \_\_\_\_\_\_ 2020 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

Львів – 2020

**Мета**​: ознайомитися із Maven та Spring.

Хід роботи

**Завдання No 1.**

Використовуючи технологію Spring створити веб-сервер, який повинен

рахувати вартість туру.

Завдання:

1. Створити структуру класів, які відповідатимуть завданню.

2. Дані можна зберігати in-memory або у файлі.

3. Реалізувати мінімум 5 кінцевих точок (endpoints), які будуть

працювати відповідно до завдання. Наприклад, “/get-all-cities”,

“/calculate-tour”, “/get-all-users”.

4. Створити глобальний ExceptionHandler та використати unchecked

exception із лабораторної роботи No3.

5. Реалізувати перевірку коректності введених даних користувачем

(валідацію) там де це необхідно.

6. Допускається вибір власної теми, крім “інтернет-магазину”

**Код файлу Lab4Application.java**

package lab4.lab4;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Lab4Application {

public static void main(String[] args) {

SpringApplication.run(Lab4Application.class, args);

}

}

**Код файлу GlobalErrorHandler.java**

package lab4.lab4;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.http.HttpStatus;

import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

import org.springframework.web.bind.annotation.RestControllerAdvice;

@RestControllerAdvice

public class GlobalErrorHandler extends ResponseEntityExceptionHandler {

@ExceptionHandler({ HttpError.class })

public ResponseEntity<String> handleHttpError(HttpError e) {

return new ResponseEntity<>("Http error: " + e.getMessage(), e.getStatus());

}

@ExceptionHandler({ Exception.class })

public ResponseEntity<String> handleSimpleError(Exception e) {

return new ResponseEntity<>("Error: " + e.getMessage(), HttpStatus.INTERNAL\_SERVER\_ERROR);

}

}

**Код файлу HttpError.java**

package lab4.lab4;

import org.springframework.http.HttpStatus;

public class HttpError extends RuntimeException {

private final HttpStatus status;

public HttpError(String message, HttpStatus status) {

super(message);

this.status = status;

}

public HttpStatus getStatus() {

return this.status;

}

}

**Код файлу Book.java**

package lab4.lab4;

import org.springframework.http.HttpStatus;

import java.util.UUID;

public class Book {

private Integer price;

private final String name;

private final String itemId;

private final String authorName;

Book(String authorName, String name, int price) {

this.name = name;

this.price = price;

this.authorName = authorName;

this.itemId = UUID.randomUUID().toString();

}

public Integer getPrice() {

return price;

}

public String getName() {

return name;

}

public String getAuthorName() {

return authorName;

}

public String getItemId() {

return itemId;

}

public void setPrice(Integer price) {

if (price <= 0) {

throw new HttpError("Price must be great than 0", HttpStatus.BAD\_REQUEST);

}

this.price = price;

};

public void validate() {

if (!(this.price instanceof Integer && price > 0)) {

throw new HttpError("Price must be integer and great than 0", HttpStatus.BAD\_REQUEST);

}

if (!(this.name instanceof String && this.name.length() > 0)) {

throw new HttpError("Invalid book name", HttpStatus.BAD\_REQUEST);

}

if (!(this.authorName instanceof String && this.authorName.length() > 0)) {

throw new HttpError("Invalid book author name", HttpStatus.BAD\_REQUEST);

}

}

}

**Код файлу BookRoutes.java**

package lab4.lab4;

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

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

@RestController

@RequestMapping("/books")

public class BookRoutes {

@Autowired

private BookMethods bookMethods;

@PostMapping

void create(@RequestBody Book book) {

bookMethods.createNew(book);

}

@GetMapping

Book[] getAll() {

return bookMethods.getAll();

}

@GetMapping("/{itemId}")

Book getById(@PathVariable("itemId") String itemId) {

return bookMethods.getById(itemId);

}

@PutMapping("/{itemId}")

void updateById(@PathVariable("itemId") String itemId, @RequestBody Book book) {

bookMethods.updateById(itemId, book);

}

@DeleteMapping("/{itemId}")

void deleteById(@PathVariable("itemId") String itemId) {

bookMethods.deleteById(itemId);

}

}

**Код файлу BookMethods.java**

package lab4.lab4;

import org.springframework.stereotype.Service;

import org.springframework.http.HttpStatus;

import java.util.ArrayList;

@Service

public class BookMethods {

private ArrayList<Book> currentBooks = new ArrayList<>();

public void createNew(Book book) {

book.validate();

currentBooks.add(book);

}

public Book[] getAll() {

return currentBooks.toArray(new Book[currentBooks.size()]);

}

public Book getById(String itemId) {

for(Book book : currentBooks) {

if(book.getItemId().equals(itemId)) {

return book;

}

}

throw new HttpError("No book found", HttpStatus.BAD\_REQUEST);

}

public void updateById(String itemId, Book updatedBook) {

Book book = this.getById(itemId);

book.setPrice(updatedBook.getPrice());

}

public Book deleteById(String itemId) {

for(int i = 0; i < currentBooks.size(); i++) {

if(currentBooks.get(i).getItemId().equals(itemId)) {

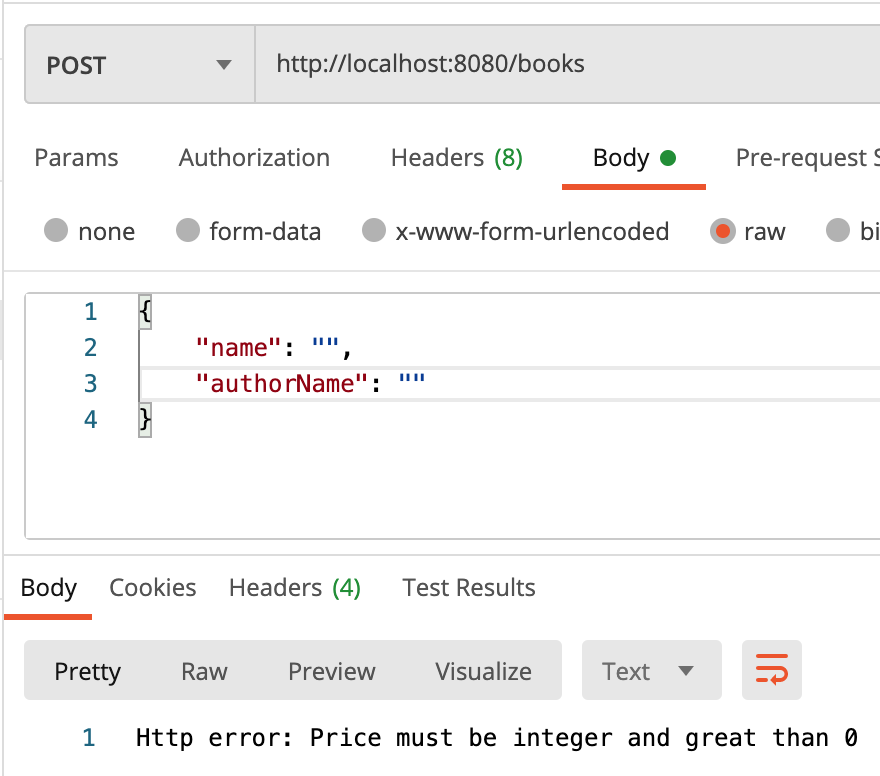
return currentBooks.remove(i);

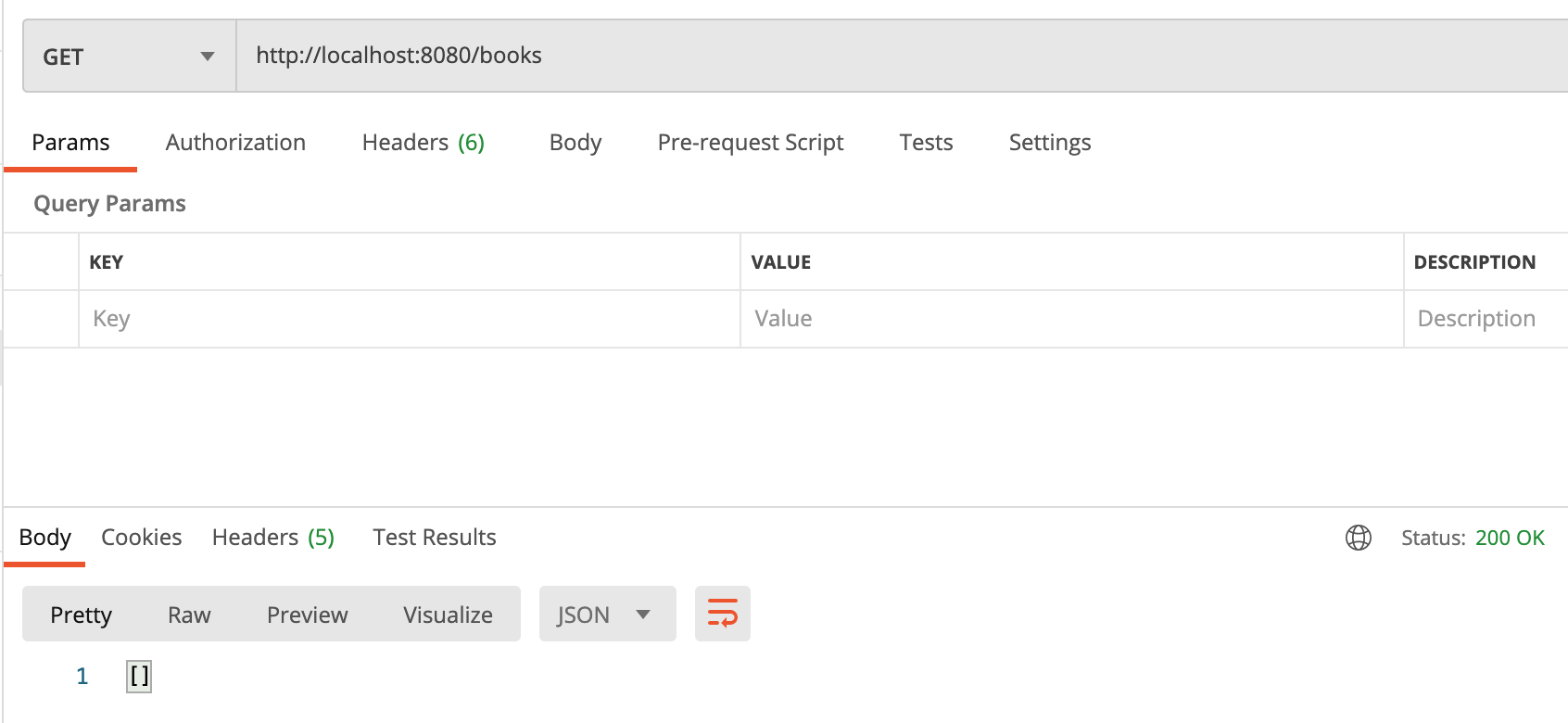
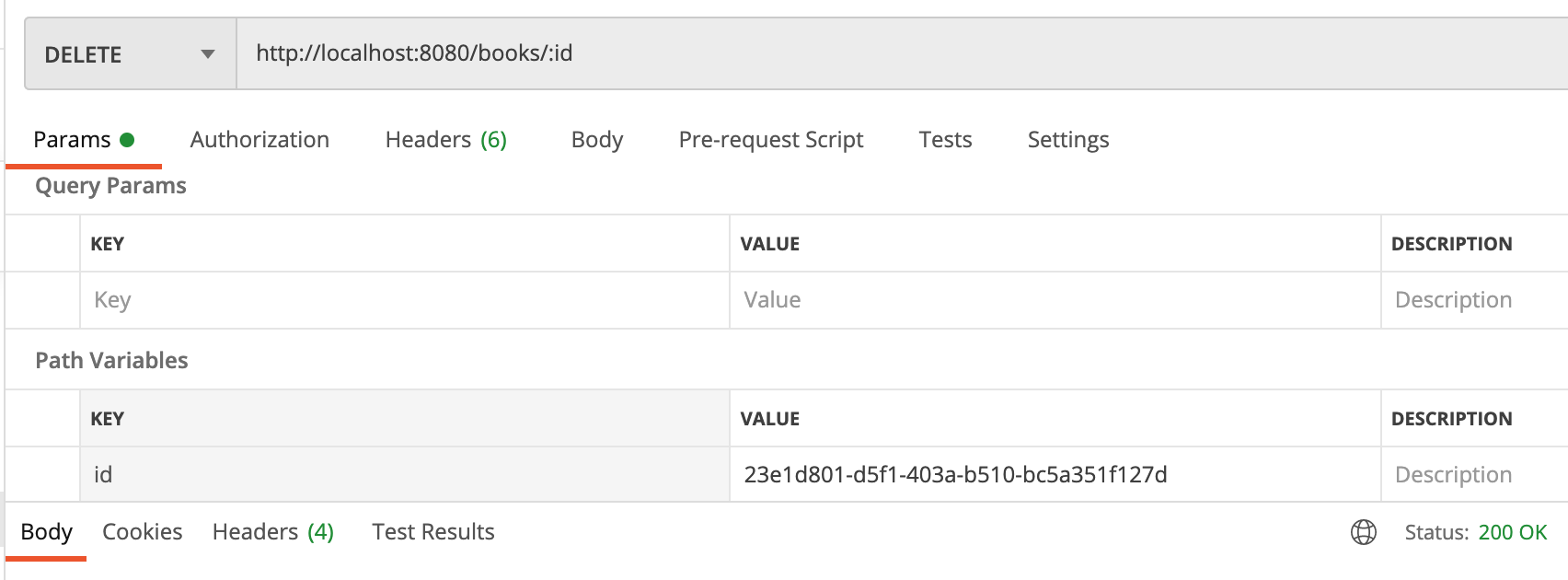
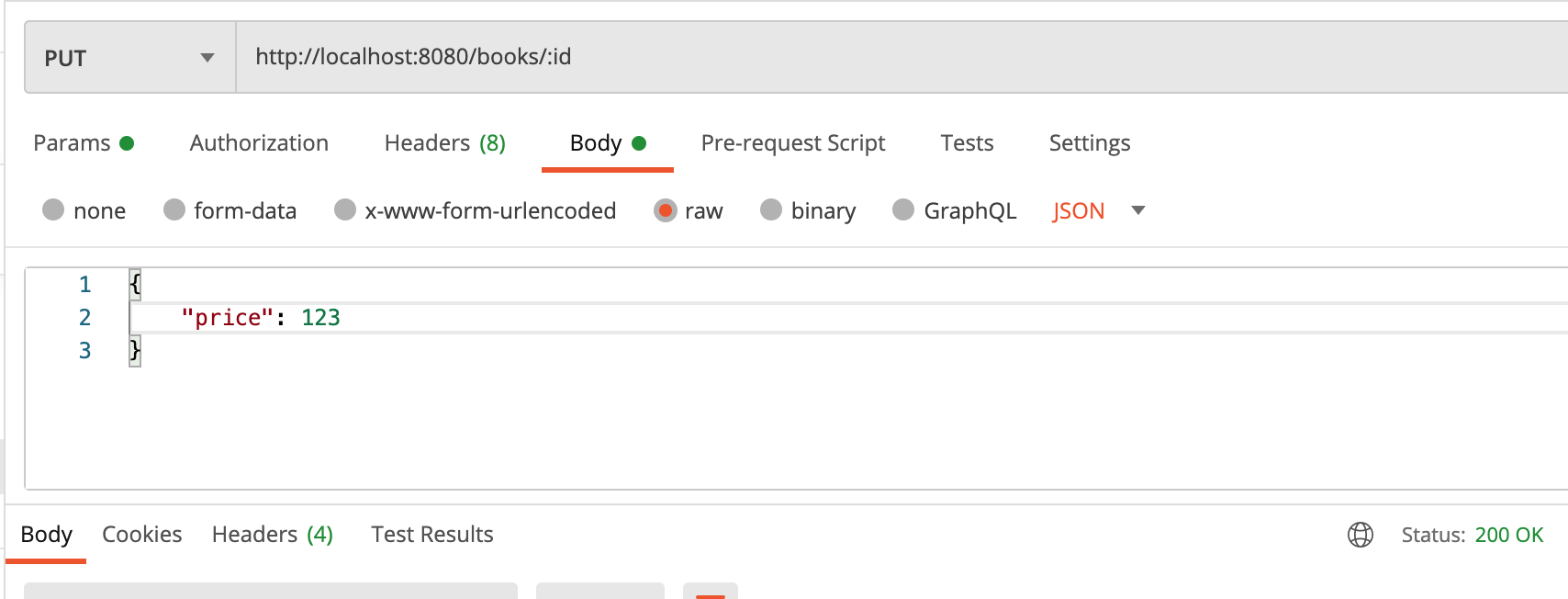
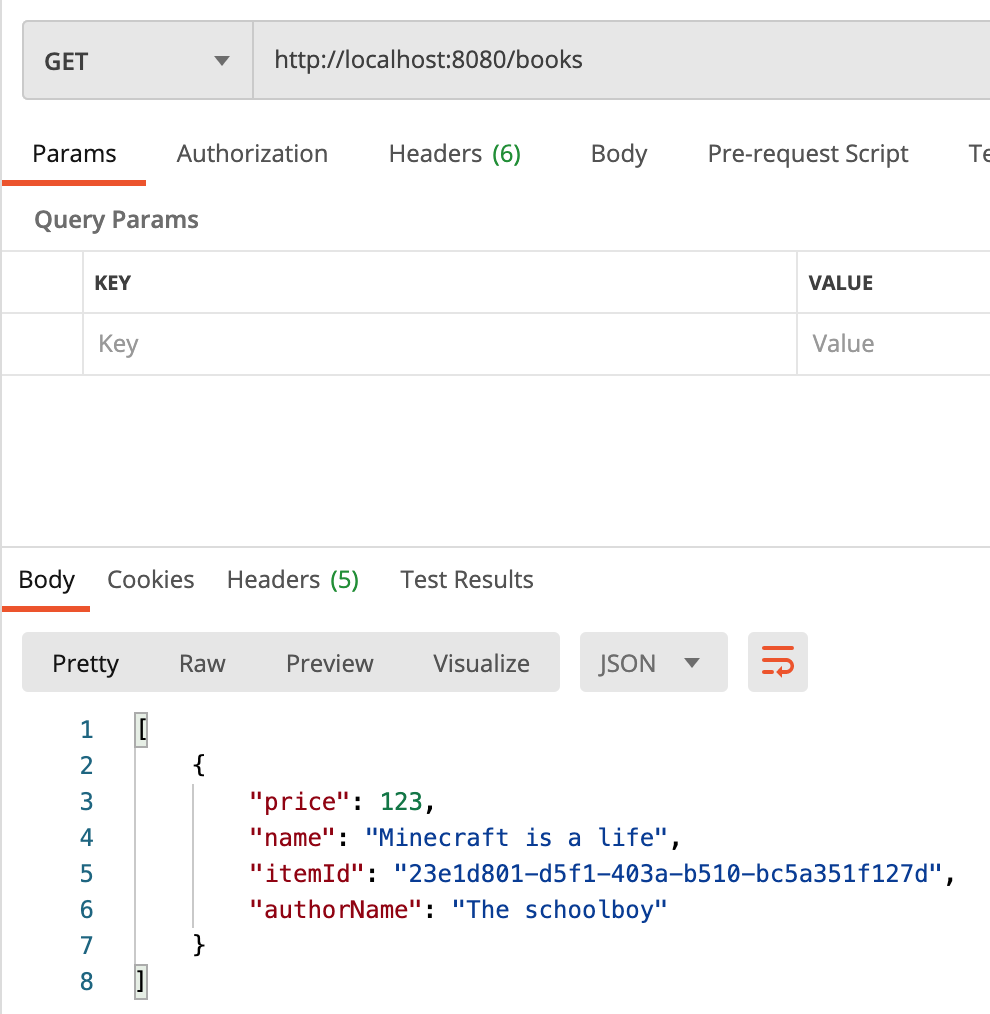
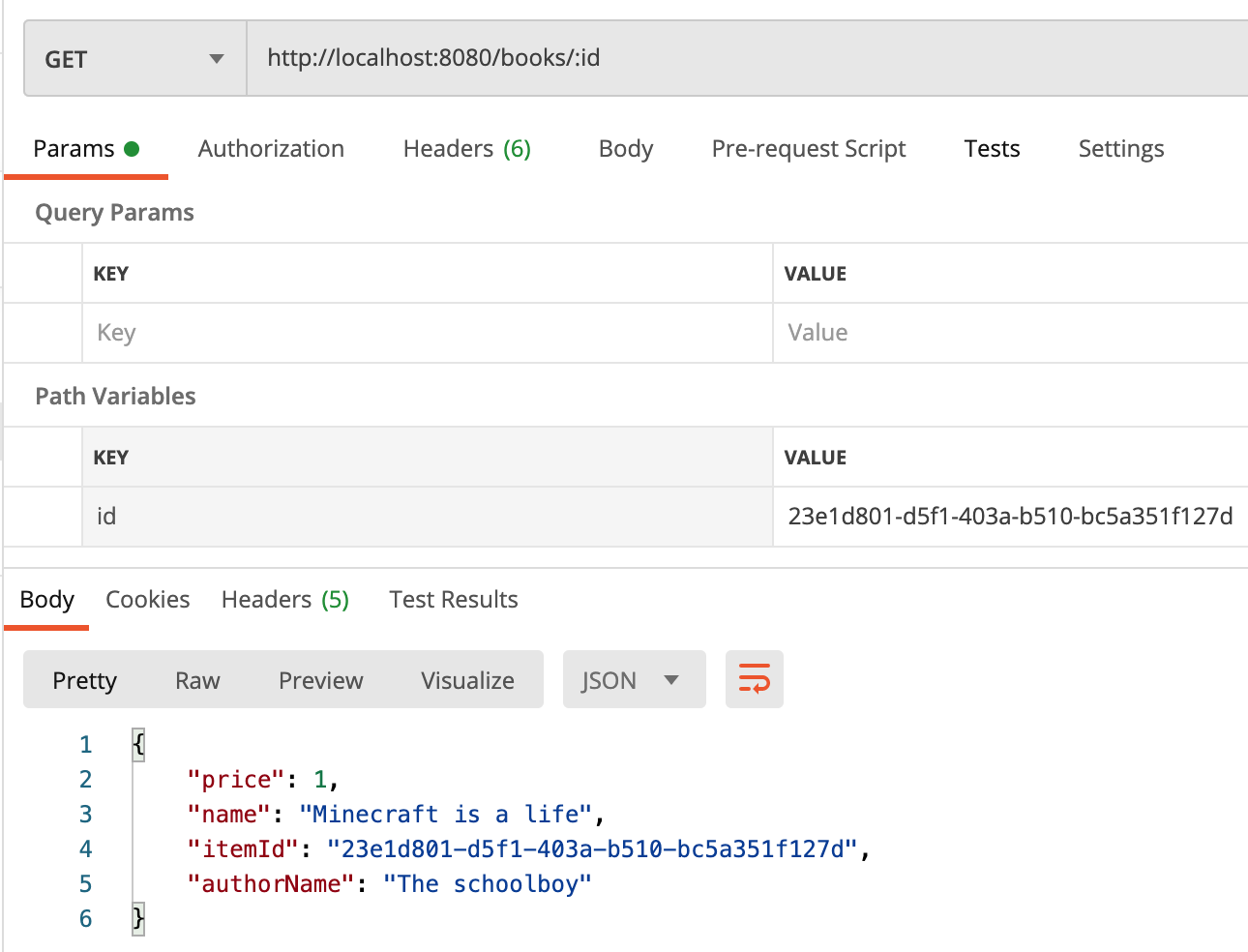
}

}

throw new HttpError( "No book found", HttpStatus.BAD\_REQUEST);

}

}****

****

**Висновок:** на даній лабораторній роботі я ознайомився із Maven та Spring..