## 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

Гук П. М.

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 CinemaApplication.java**

package lab4.cinema;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class CinemaApplication {

public static void main(String[] args) {

SpringApplication.run(CinemaApplication.class, args);

}

}

**Код файлу CustomGlobalExceptionHandler.java**

package lab4.cinema;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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

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

@RestControllerAdvice

public class CustomGlobalExceptionHandler extends ResponseEntityExceptionHandler {

@ExceptionHandler({ HttpException.class })

public ResponseEntity<String> handleHttpError(HttpException e) {

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

}

@ExceptionHandler({ Exception.class })

public ResponseEntity<String> handleError(Exception e) {

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

}

}

**Код файлу HttpException.java**

package lab4.cinema;

import org.springframework.http.HttpStatus;

public class HttpException extends RuntimeException {

private final HttpStatus status;

public HttpException(HttpStatus status, String errorMessage) {

super(errorMessage);

this.status = status;

}

public HttpStatus getStatus() {

return this.status;

}

}

**Код файлу Movie.java**

package lab4.cinema;

import org.springframework.http.HttpStatus;

import java.util.UUID;

public class Movie {

private final String id;

private String name;

private Integer rating;

Movie(String name, int rating) {

this.name = name;

this.rating = rating;

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

}

public String getName() {

return name;

}

public Integer getRating() {

return rating;

}

public void setName(String name) {

this.name = name;

}

public void setRating(Integer rating) {

this.rating = rating;

}

public String getId() {

return id;

}

public void validate() {

boolean isNameValid = this.name instanceof String && this.name.length() != 0;

boolean isRatingValid = this.rating instanceof Integer && this.rating >= 0 && this.rating <= 5;

if (!isNameValid) {

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

}

if (!isRatingValid) {

throw new HttpException(HttpStatus.BAD\_REQUEST, "Invalid rating");

}

}

}

**Код файлу MovieController.java**

package lab4.cinema;

import org.springframework.http.HttpStatus;

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

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

@RestController

@RequestMapping("/movies")

public class MovieController {

@Autowired

private MovieService movieService;

@PostMapping

@ResponseStatus(HttpStatus.CREATED)

void createMovie(@RequestBody Movie movie) {

movie.validate();

movieService.save(movie);

}

@GetMapping

Movie[] getAllMovies() {

return movieService.getAll();

}

@GetMapping("/{movieId}")

Movie getMovieById(@PathVariable("movieId") String movieId) {

return movieService.getById(movieId);

}

@PutMapping("/{movieId}")

void updateMovieById(@PathVariable("movieId") String movieId, @RequestBody Movie movie) {

movie.validate();

movieService.updateById(movieId, movie);

}

@DeleteMapping("/{movieId}")

void deleteMovieById(@PathVariable("movieId") String movieId) {

movieService.deleteById(movieId);

}

}

**Код файлу MovieService.java**

package lab4.cinema;

import org.springframework.http.HttpStatus;

import org.springframework.stereotype.Service;

import java.util.ArrayList;

@Service

public class MovieService {

private ArrayList<Movie> movies = new ArrayList<>();

public void save(Movie movie) {

movies.add(movie);

}

public Movie[] getAll() {

return movies.toArray(new Movie[movies.size()]);

}

public Movie getById(String id) {

for(Movie movie : movies) {

if(movie.getId().equals(id)) {

return movie;

}

}

throw new HttpException(HttpStatus.BAD\_REQUEST, "No movie with id: " + id + " exists");

}

public void updateById(String id, Movie updatedMovie) {

Movie movie = this.getById(id);

movie.setName(updatedMovie.getName());

movie.setRating(updatedMovie.getRating());

}

public Movie deleteById(String id) {

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

if(movies.get(i).getId().equals(id)) {

return movies.remove(i);

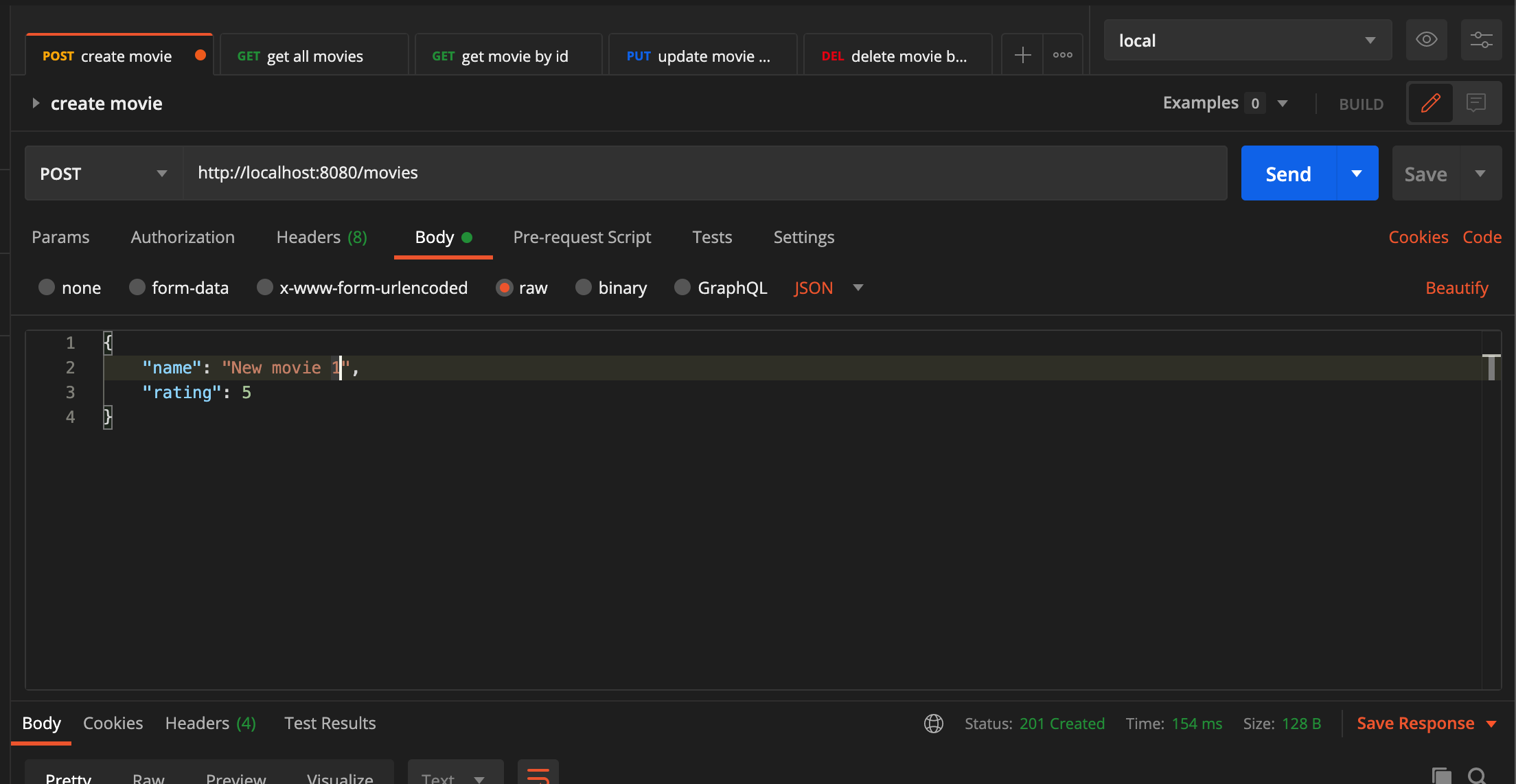
}

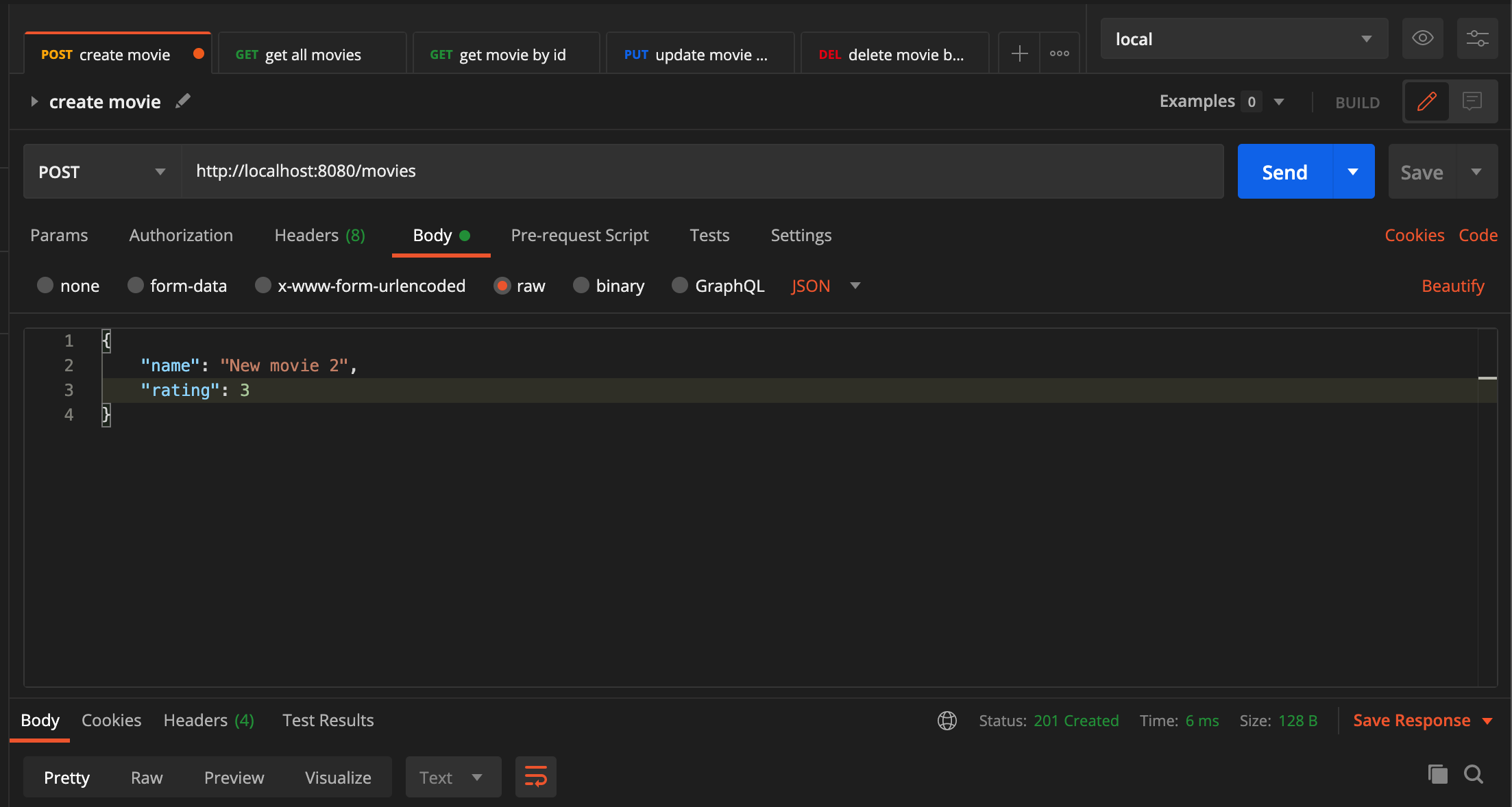
}

throw new HttpException(HttpStatus.BAD\_REQUEST, "No movie with id: " + id + " exists");

}

}

****

****

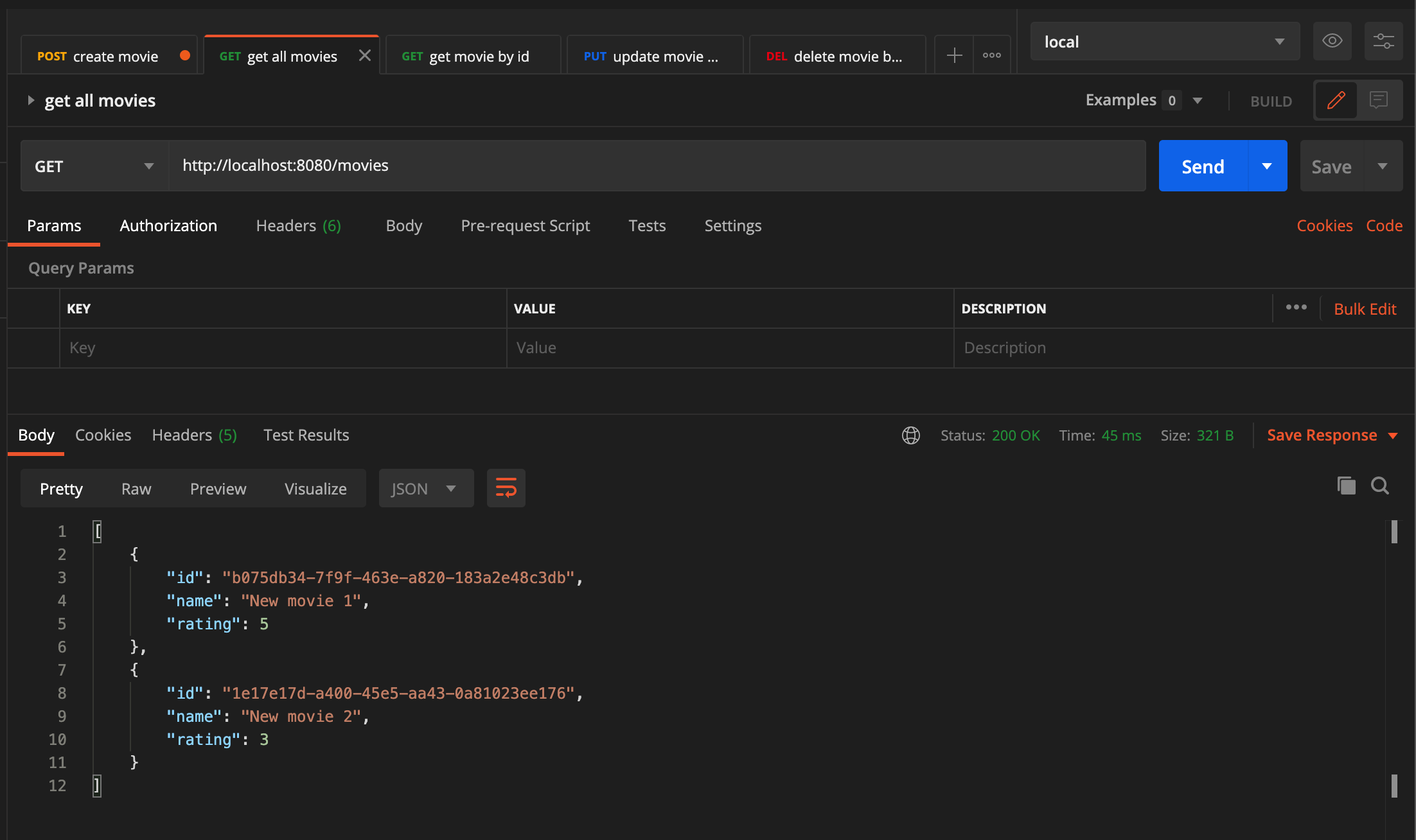
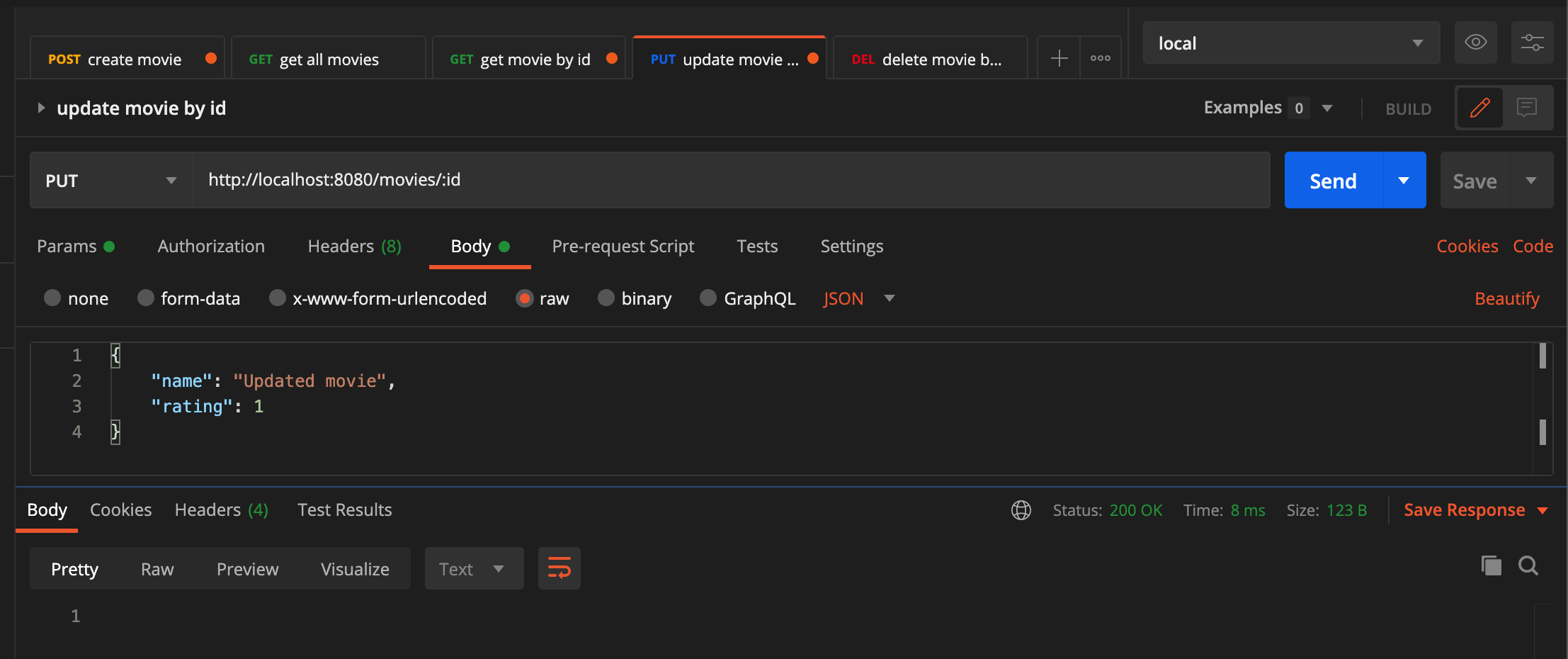
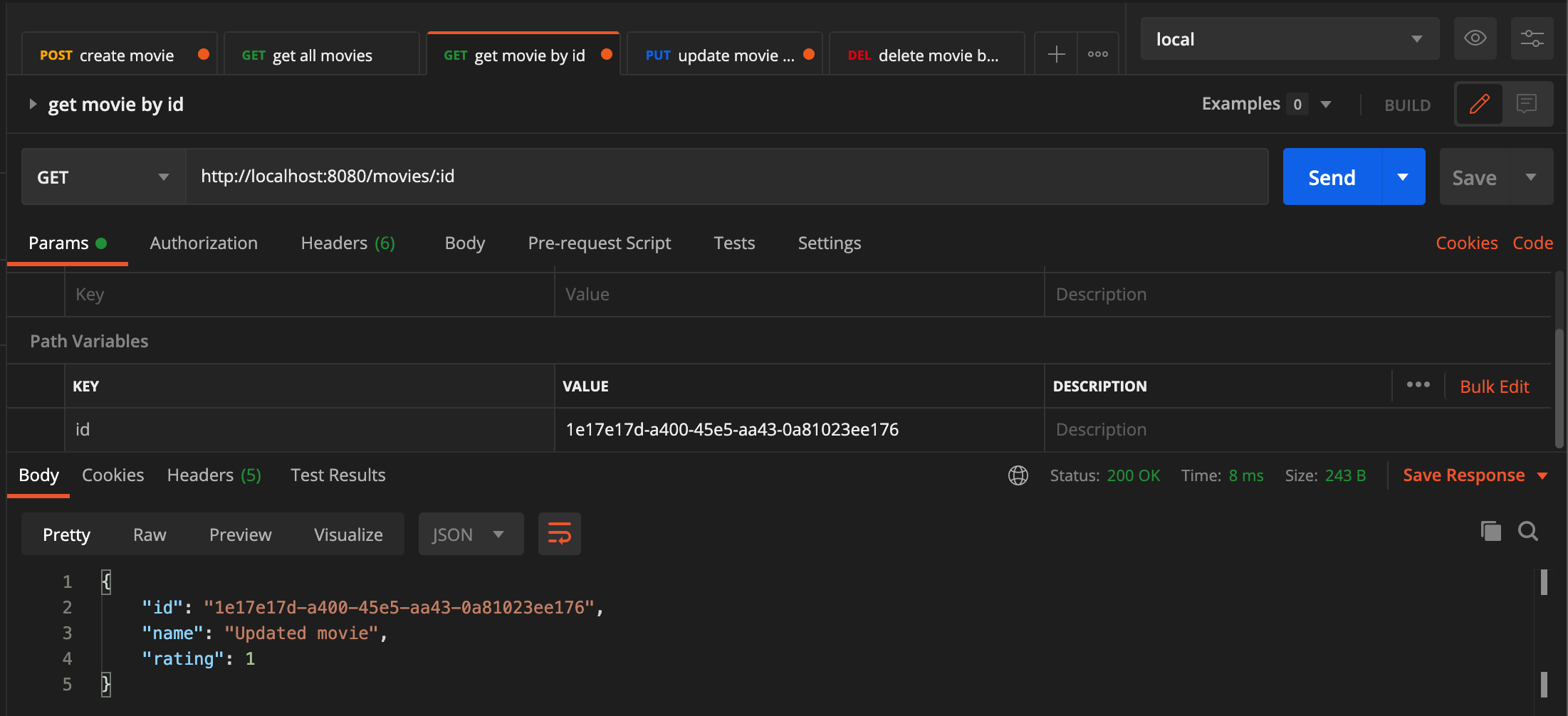
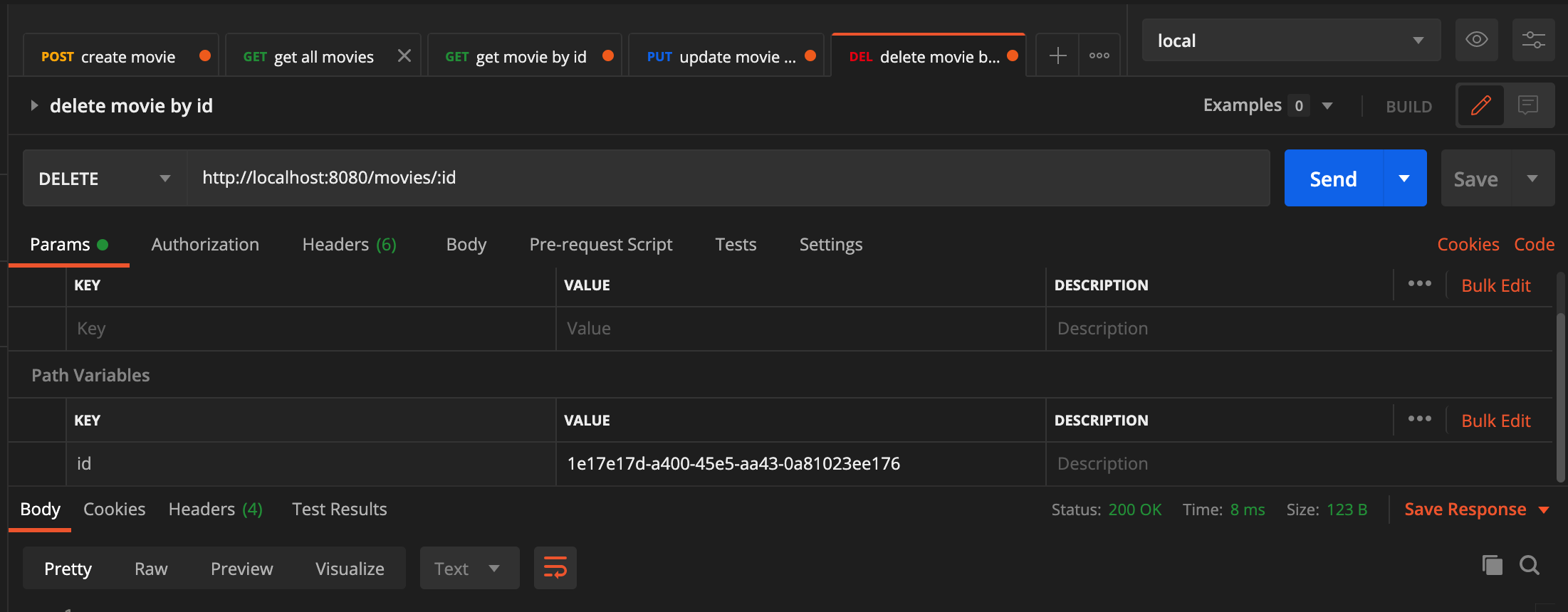
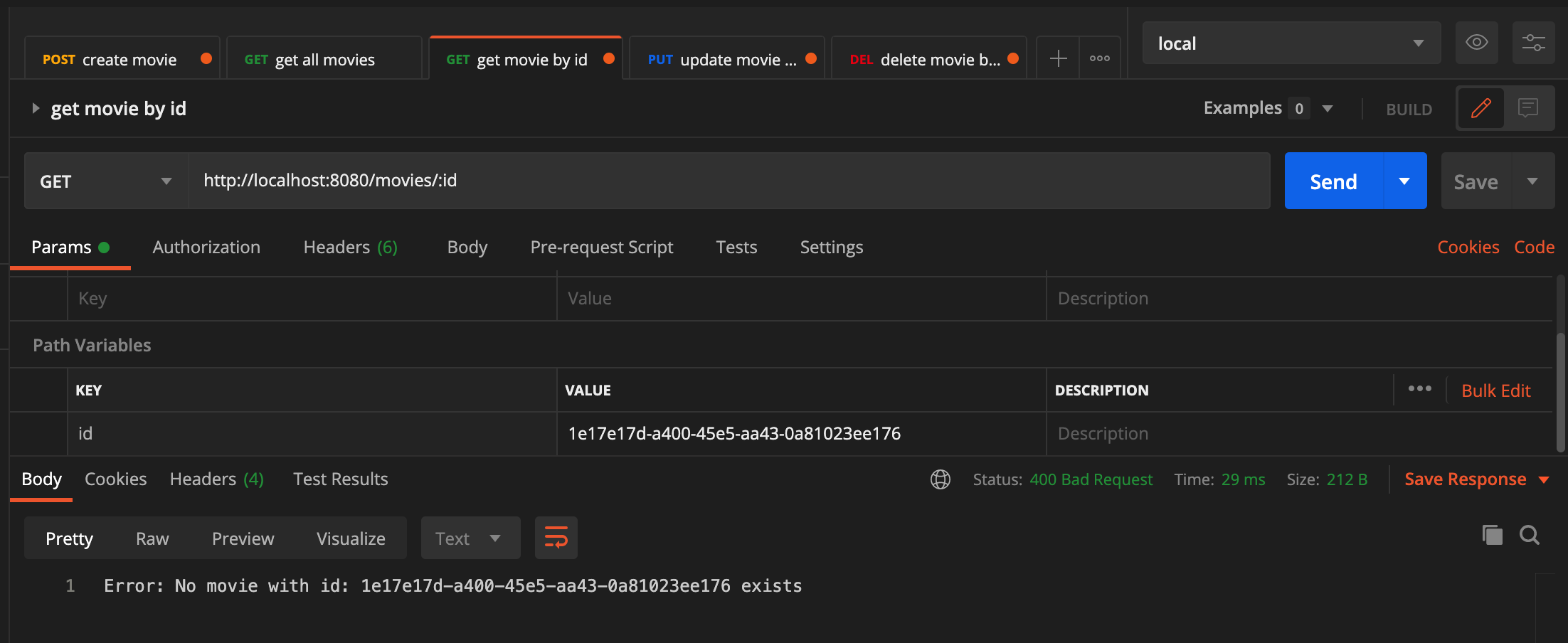
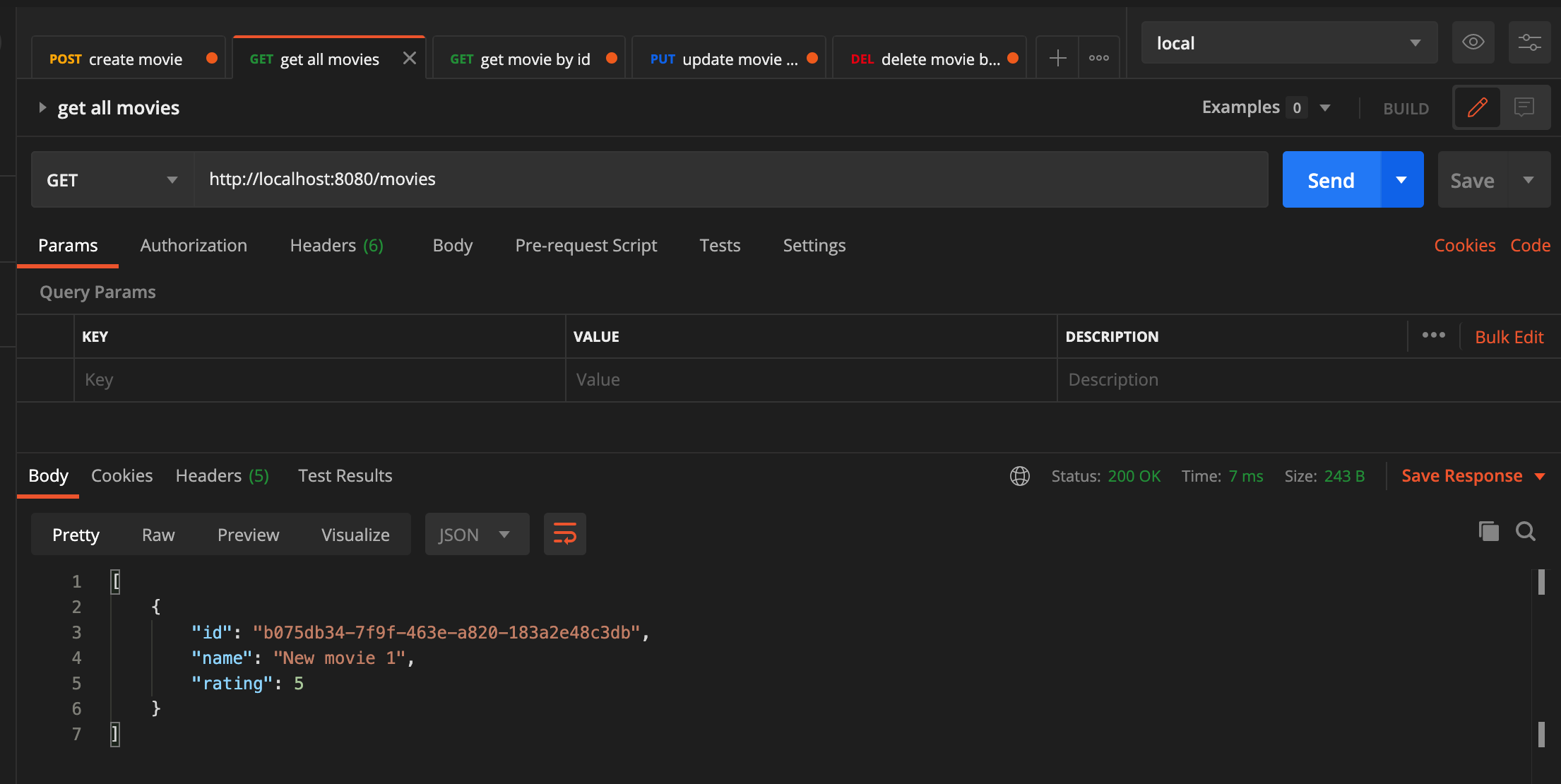
****     

Рис.1 Результат виконання роботи програми

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