**Hands On Exercise 9: Creating a Spring Boot Application**

**CODE:**

**Book Class: -**

package com.cognizant.LibraryManagement;  
import jakarta.persistence.\*;  
  
@Entity  
public class Book {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 private String title;  
  
 private String author;  
  
 private double price;  
  
 public Book() {}  
  
 public Book(String title, String author, double price) {  
 this.title = title;  
 this.author = author;  
 this.price = price;  
 }  
  
 public Long getId() { return id; }  
  
 public void setId(Long id) { this.id = id; }  
  
 public String getTitle() { return title; }  
  
 public void setTitle(String title) { this.title = title; }  
  
 public String getAuthor() { return author; }  
  
 public void setAuthor(String author) { this.author = author; }  
  
 public double getPrice() { return price; }  
  
 public void setPrice(double price) { this.price = price; }  
}

**BookRepository Interface: -**

package com.cognizant.LibraryManagement;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface BookRepository extends JpaRepository<Book, Long> {  
}

**BookController Class: -**

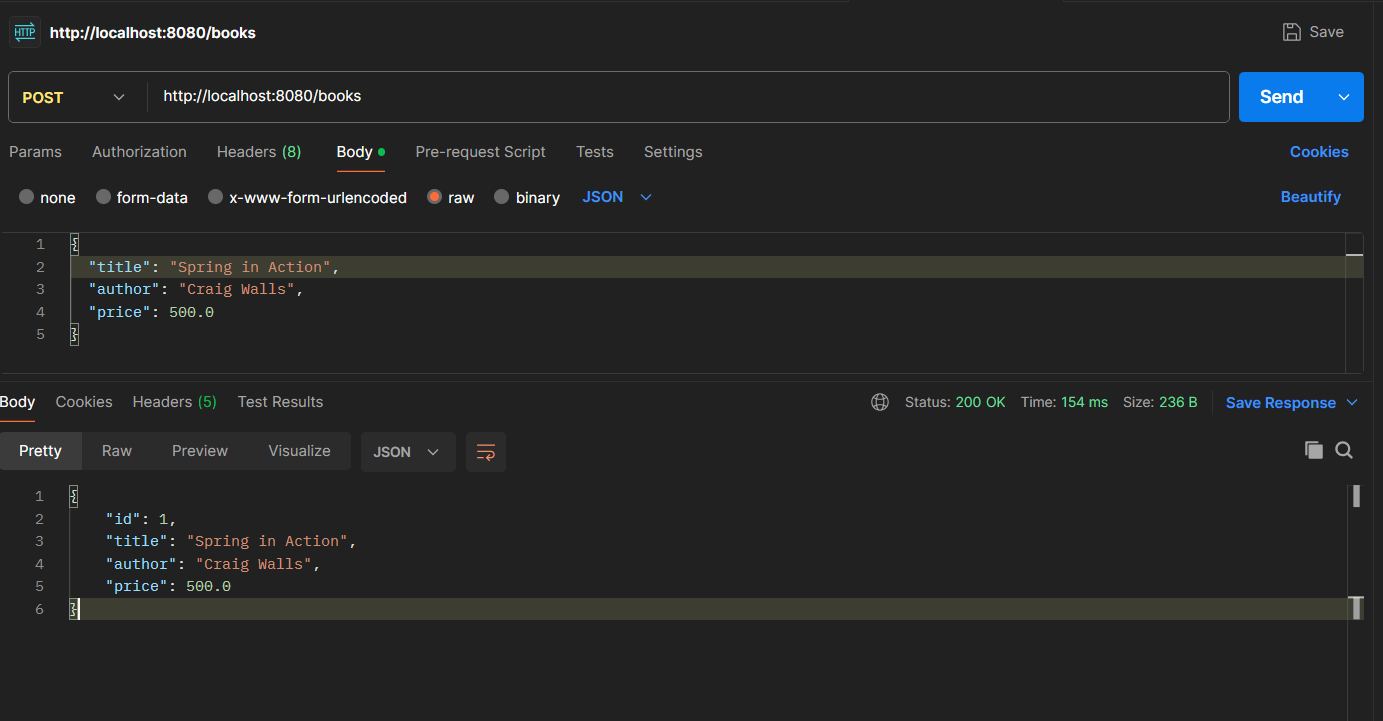
package com.cognizant.LibraryManagement;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
import java.util.Optional;  
  
@RestController  
@RequestMapping("/books")  
public class BookController {  
  
 @Autowired  
 private BookRepository bookRepository;  
  
 @PostMapping  
 public Book addBook(@RequestBody Book book) {  
 return bookRepository.save(book);  
 }  
  
 @GetMapping  
 public List<Book> getAllBooks() {  
 return bookRepository.findAll();  
 }  
  
 @GetMapping("/{id}")  
 public Optional<Book> getBookById(@PathVariable Long id) {  
 return bookRepository.findById(id);  
 }  
  
 @PutMapping("/{id}")  
 public Book updateBook(@PathVariable Long id, @RequestBody Book updatedBook) {  
 return bookRepository.findById(id).map(book -> {  
 book.setTitle(updatedBook.getTitle());  
 book.setAuthor(updatedBook.getAuthor());  
 book.setPrice(updatedBook.getPrice());  
 return bookRepository.save(book);  
 }).orElseGet(() -> {  
 updatedBook.setId(id);  
 return bookRepository.save(updatedBook);  
 });  
 }  
  
 @DeleteMapping("/{id}")  
 public String deleteBook(@PathVariable Long id) {  
 bookRepository.deleteById(id);  
 return "Successfully deleted the book with id:"+id;  
 }  
}

**Main Class: -**

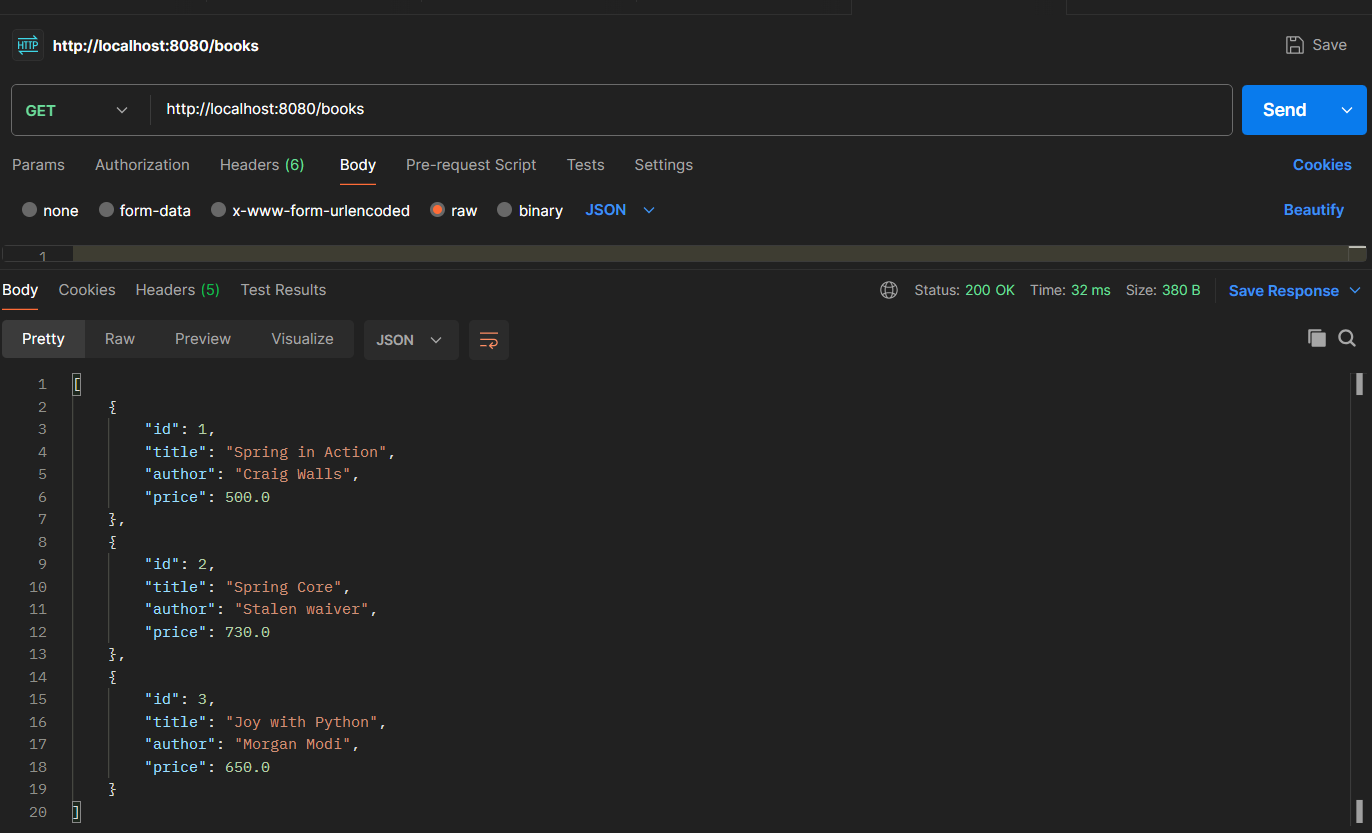
package com.cognizant.LibraryManagement;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class LibraryManagementApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(LibraryManagementApplication.class, args);  
 }  
  
}

**OUTPUT:**

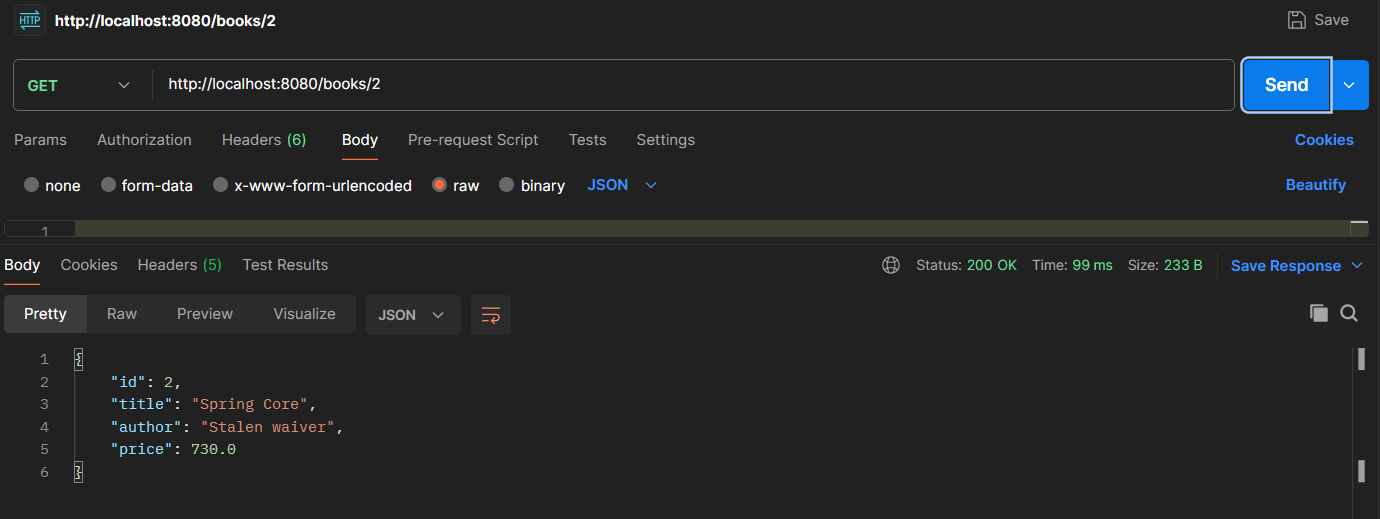
Testing POST mapping by adding books:



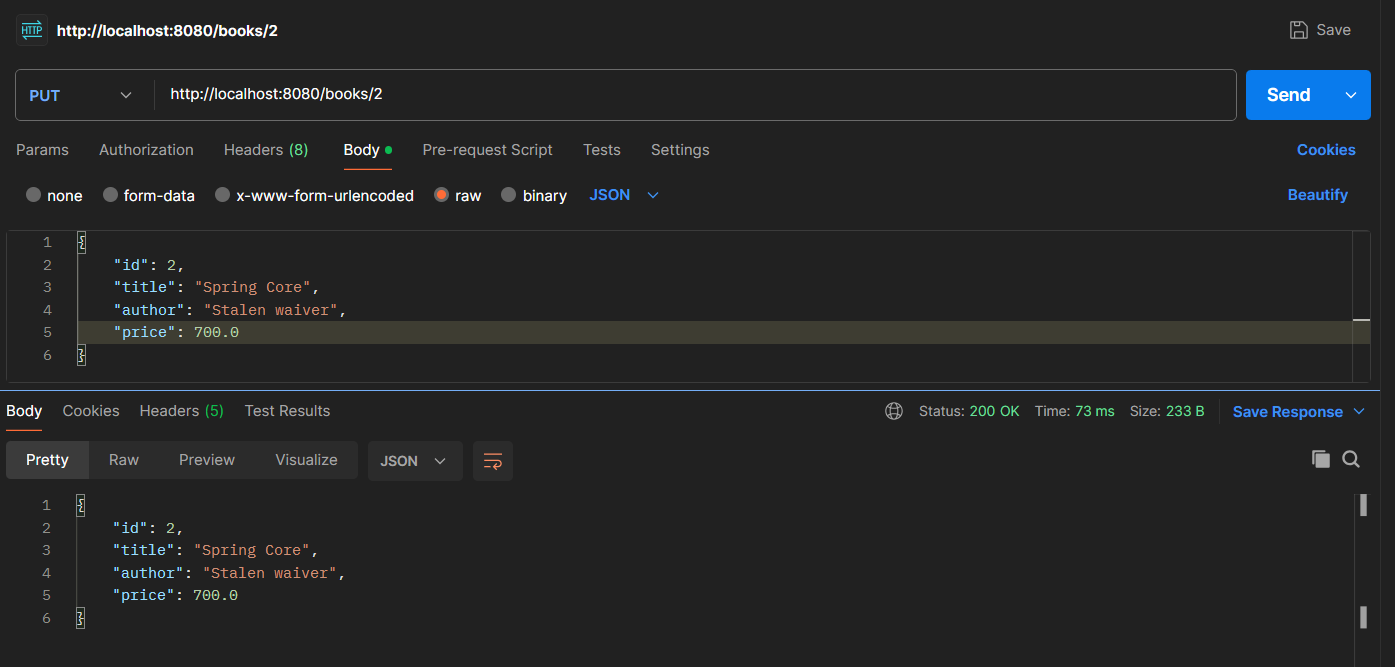
Testing GET mapping to fetch all books present in the database: -



Testing GET Mapping by giving a specific id (here I have given 2): -



Testing PUT Mapping by updating the price of the 2nd book from 730 to 700: -



Testing DELETE Mapping Method by Deleting the book with id 1: -

