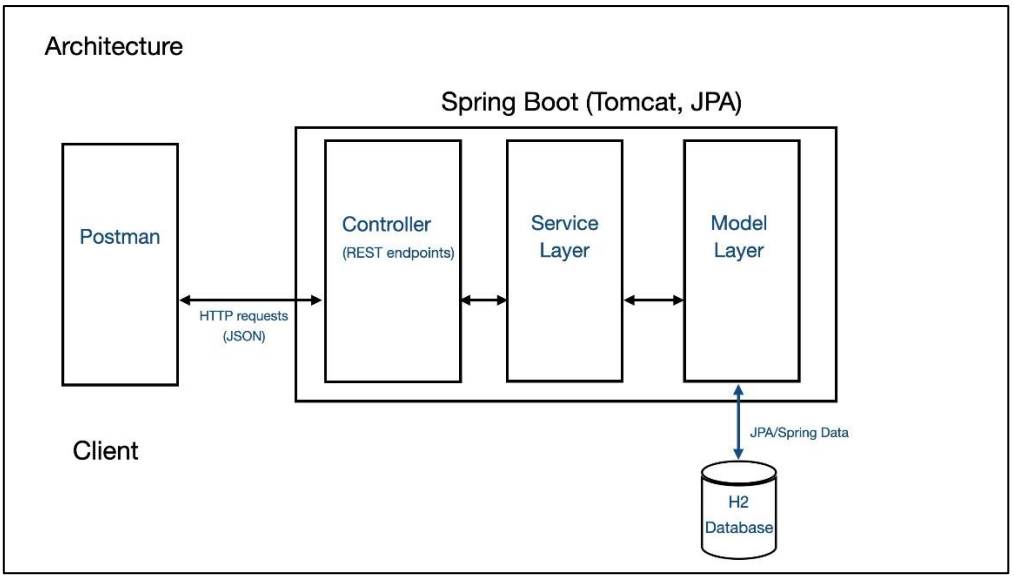
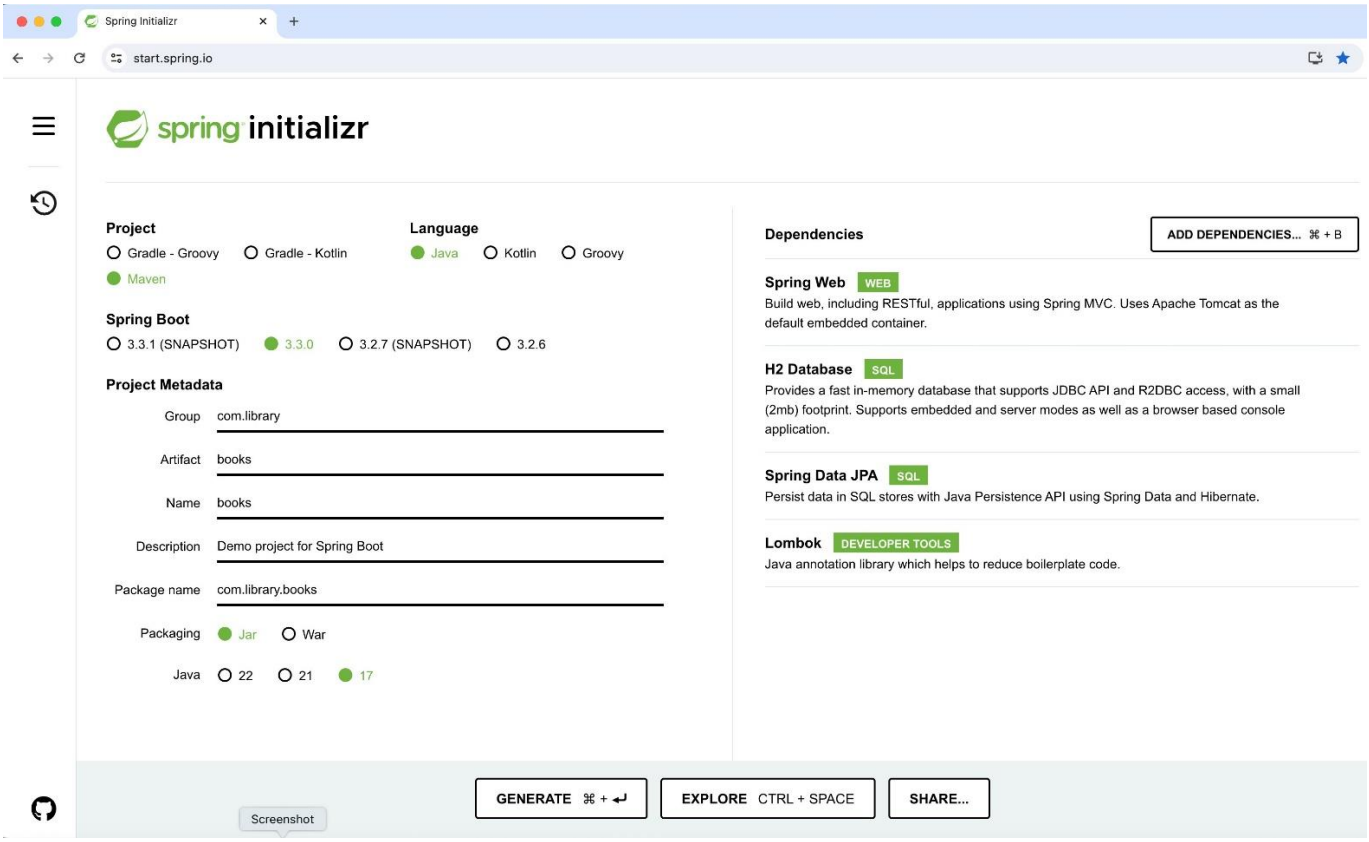
Spring Boot Assignment

In this assignment, we are going to develop a Books Spring Boot application. The architecture is similar to the one outlined in the course:



1. Go to *start.spring.io* and setup the following artefacts and application dependencies:



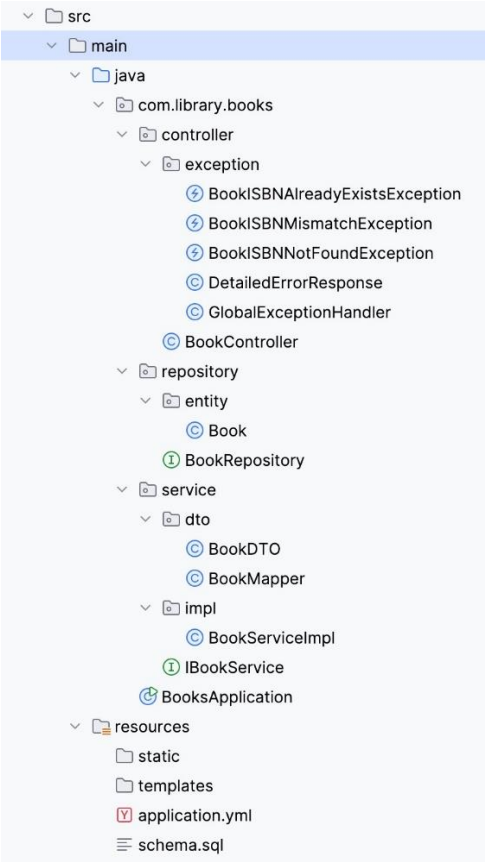
2. The *book* table in the database is created by the provided *schema.sql*.

```sql
CREATE TABLE IF NOT EXISTS `book` (
    `id` int AUTO_INCREMENT  PRIMARY KEY,
    `book_title` varchar(100) NOT NULL,
    `authors` varchar(100) NOT NULL,
    `publisher` varchar(100) NOT NULL,
    `isbn` varchar(30) NOT NULL,
    `year_published` integer NOT NULL,
    `price` integer NOT NULL
    );
```

3. The *application.yml* is also provided.

```yaml
server:
  port: 8080
spring:
  datasource:
    url: jdbc:h2:mem:testdb
    driverClassName: org.h2.Driver
    username: sa
    password: ''
  h2:
    console:
      enabled: true
  jpa:
    database-platform: org.hibernate.dialect.H2Dialect
    hibernate:
      ddl-auto: update
    show-sql: true
```

4. Packages overview:

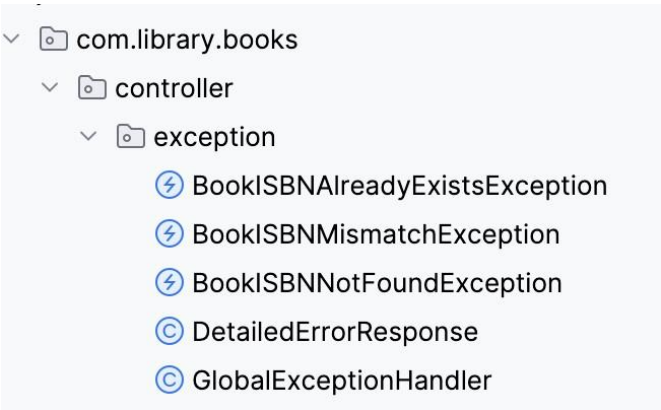


5. The URI/verb design. with associates semantics, is as follows:
   a. http://localhost:8080/books
      i. GET - retrieve all books (from the database)
      ii. POST - add a book
      iii. DELETE - delete all books
      iv. PUT - not allowed on collection resources (e.g. /books)
      v. OPTIONS - *Allow* header to return HEAD, GET, POST and DELETE.
   b. http://localhost:8080/books/{isbn}
      i. GET - retrieve the book identified by the isbn provided
      ii. POST - not allowed
      iii. DELETE - delete that particular book
      iv. PUT - update that book with the details from the body of the HTTP request. Note that the ISBN numbers from the URL and message body must match.
      v. OPTIONS - *Allow* header to return HEAD, GET, PUT and DELETE.

c. http://localhost:8080/books/book?authors=Sean
   i. GET - retrieve all books for a specific author - in order to code this, you will need the following method in your *BookRepository* class. Use the following imports:
      - *import org.springframework.data.jpa.repository.Query;*
      - *import org.springframework.data.repository.query.Param;*

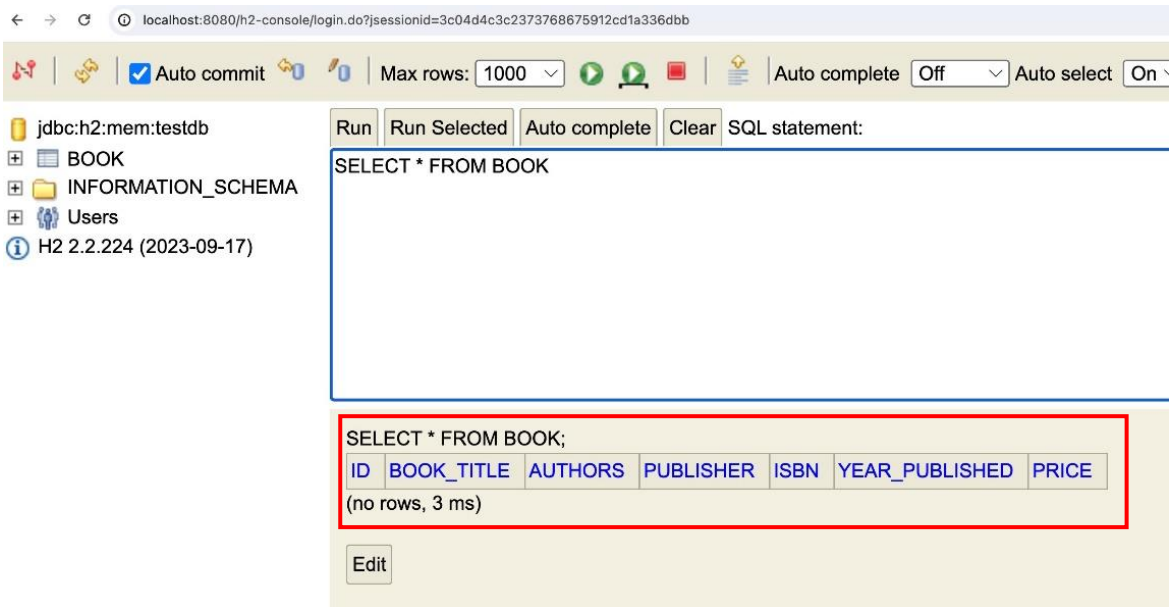
```
21    // SQL needs to be exact, would prefer a wildcard
22    @Query("select b from Book b where authors like %:authorName%")  1 usage
23    List<Book> findBooksByAuthorName(@Param("authorName") String authorName);
```

6. Exceptions - please cater for the following exceptions using a global exception controller:
   a. If you are trying to insert a book (POST) where the ISBN already exists in the database, generate a *BookISBNAlreadyExistsException*.
   b. If you are trying to update a books details (PUT) and the ISBN in the URL does not match the ISBN from the request/entity body, generate a *BookISBNMismatchException*.
   c. If you trying to delete (DELETE) or retrieve (GET) a book where the ISBN does not exist in the database then generate a *BookISBNNotFoundException*.

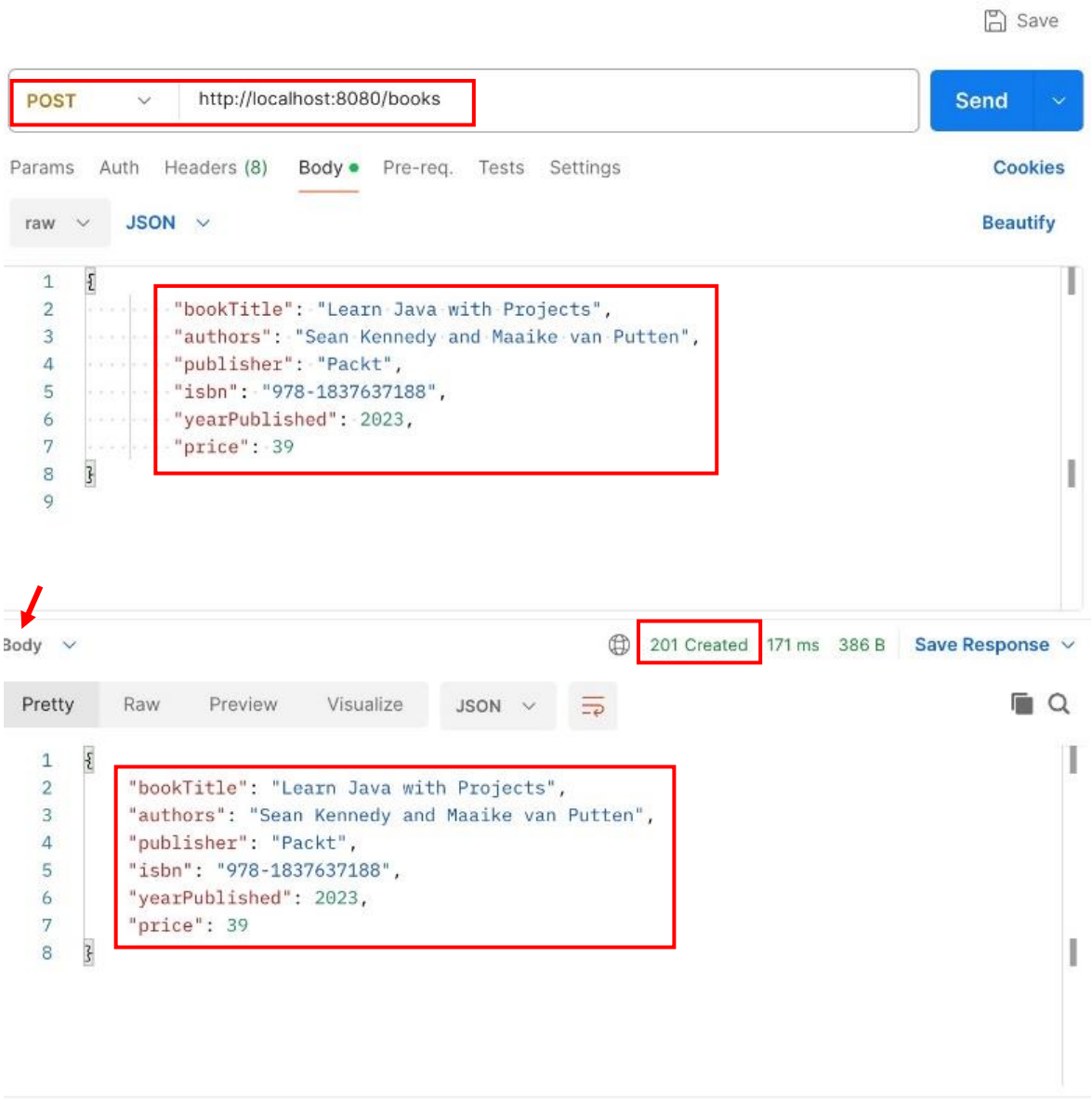


7. The following is a high level overview of what we are going to do:
    a. add a book
    b. add the exact same book again - an exception
    c. add a second (different) book
    d. retrieve all books
    e. retrieve one book (by ISBN)
    f. retrieve a book using an invalid ISBN - an exception
    g. remove one book
    h. remove a book using an invalid ISBN - an exception
    i. remove all books
    j. change “Maiike van Putten” to “MvP” for the book with ISBN 978-1837637188
    k. try to update a book when the ISBN in the URI is not the same as the ISBN in the entity body - an exception
    l. using the query/request parameter “author”, retrieve all books authored by “Sean Kennedy”
    m. try to add a book using the URI /books/{isbn} - HTTP Method Not Allowed
    n. try to update a book by using the URI /books - HTTP Method Not Allowed
    o. send a request to the collection URI to find out what verbs are supported
    p. do the same for an individual resource

8. Sample Postman screenshots:
    a. Database after starting the application (database should be empty).



b. Add first Book (1 of 2).



c. Add first book (2 of 2).

Save

POST http://localhost:8080/books Send

Params Auth Headers (8) Body Pre-req. Tests Settings Cookies

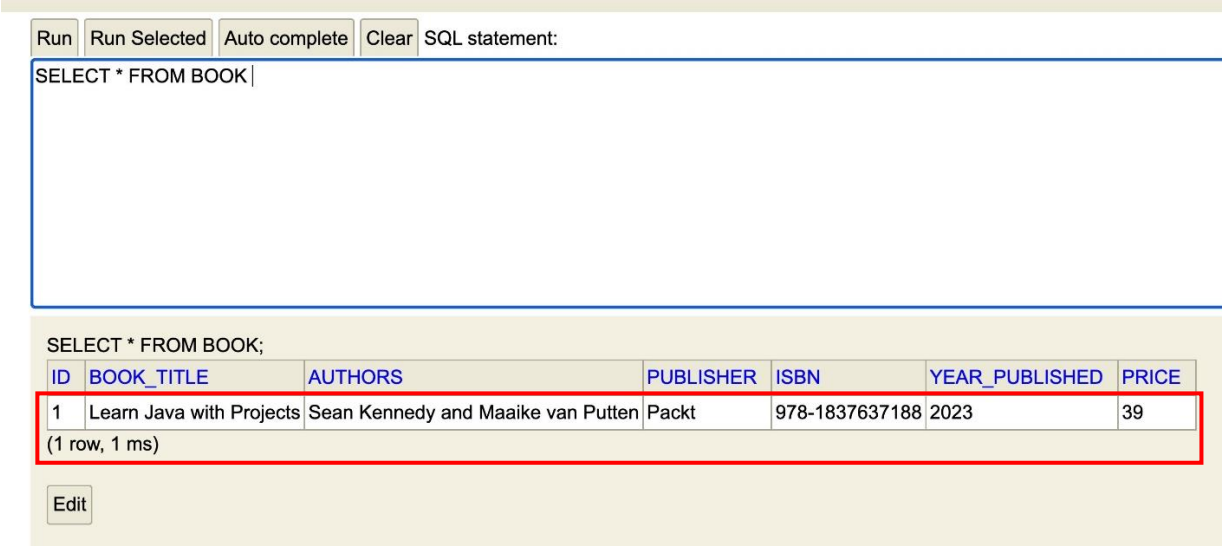
raw JSON Beautify

```
1 {
2     "bookTitle": "Learn Java with Projects",
3     "authors": "Sean Kennedy and Maaike van Putten",
4     "publisher": "Packt",
5     "isbn": "978-1837637188",
6     "yearPublished": 2023,
7     "price": 39
8 }
9
```

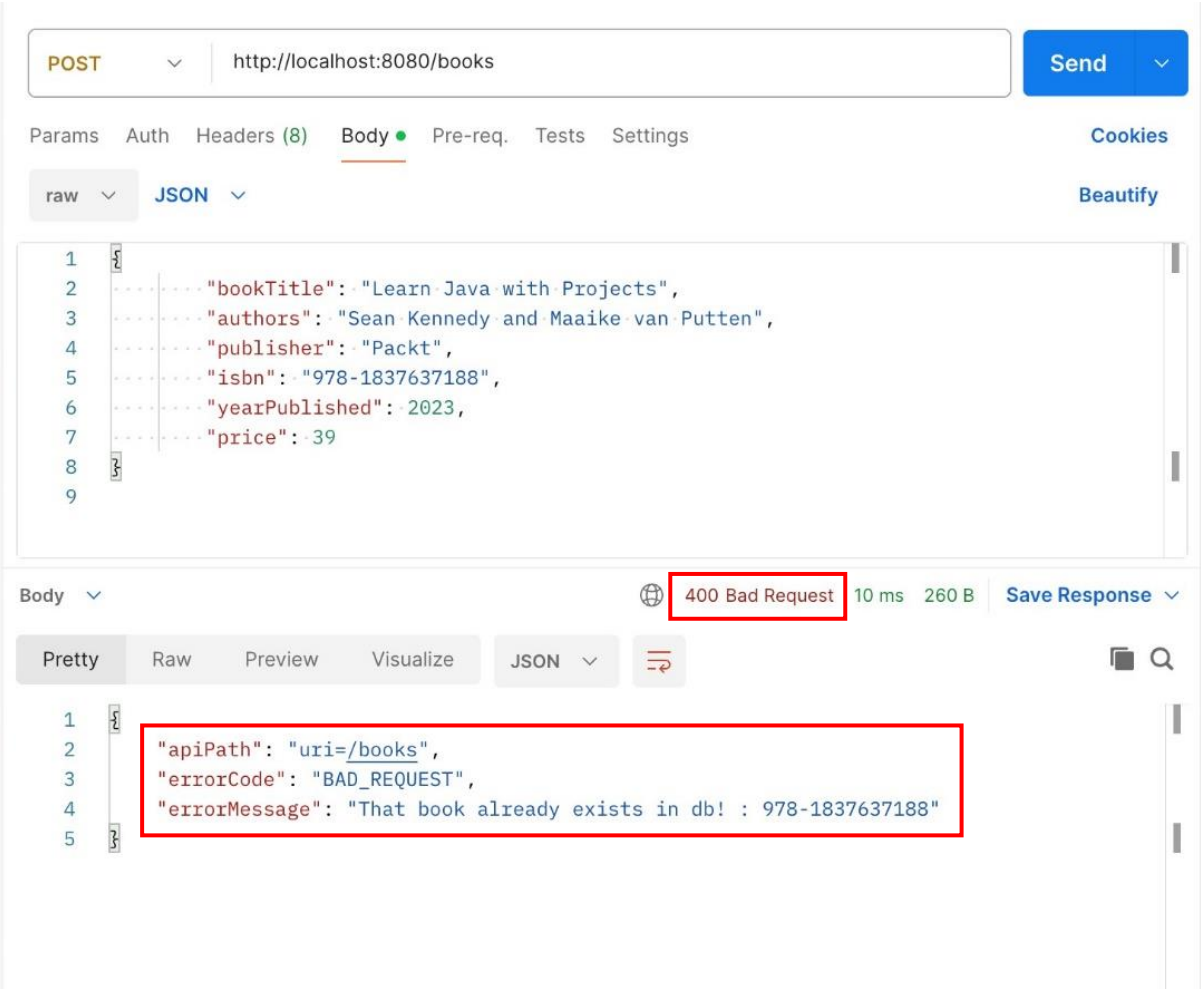
Headers 201 Created 171 ms 386 B Save Response

Key	Value
Location	http://localhost:8080/books/978-1837637188
Content-Type	application/json
Transfer-Encoding	chunked
Date	Wed, 12 Jun 2024 15:36:36 GMT
Keep-Alive	timeout=60
Connection	keep-alive

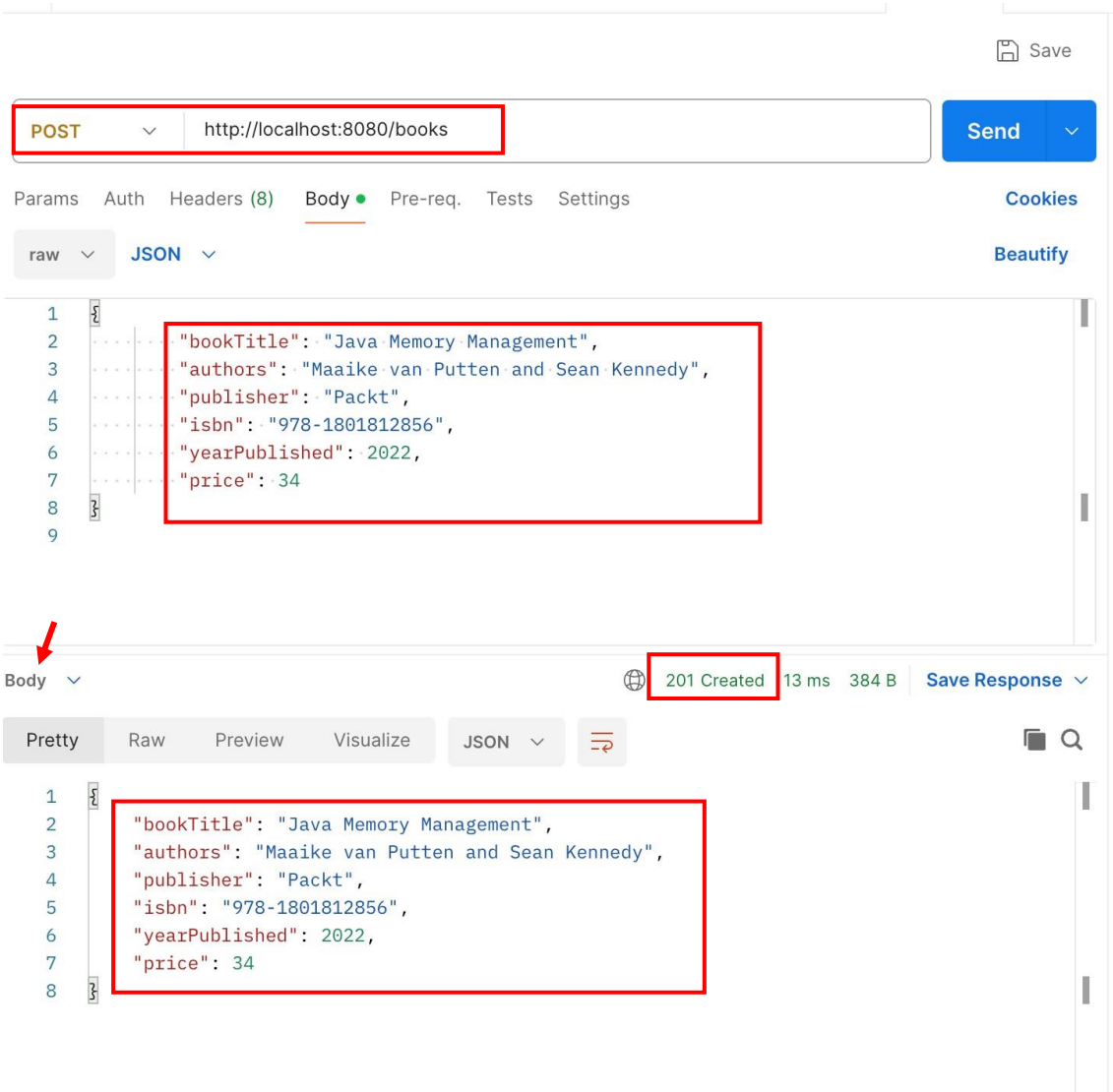
d. Database after first book added.



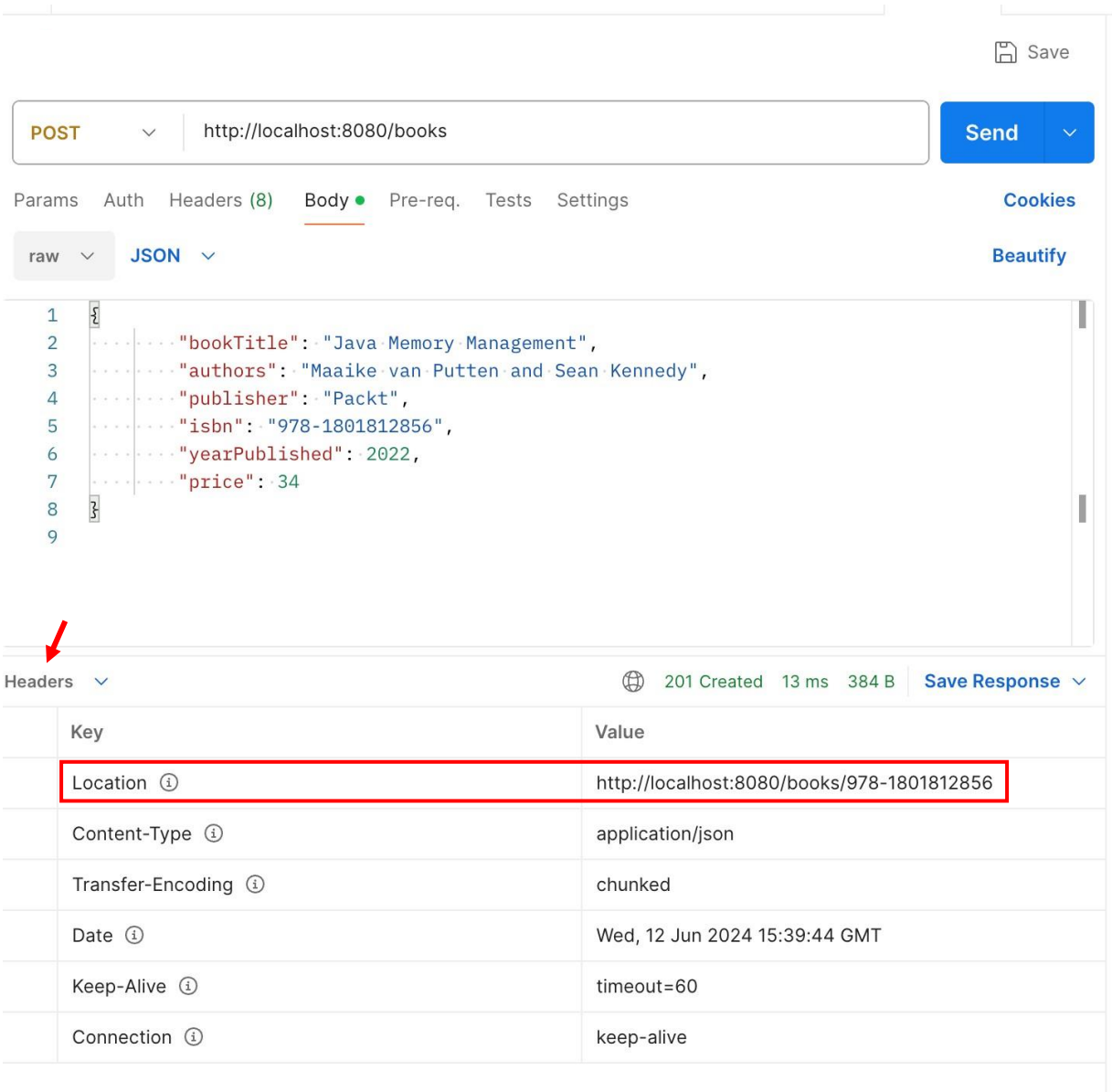
e. Attempting to add a book but the ISBN number already exists (same POST as in previous steps).



f. Add second book (1 of 2).



g. Add second book (2 of 2).



h. Database after second book added.

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM BOOK
```

SELECT * FROM BOOK;

ID	BOOK_TITLE	AUTHORS	PUBLISHER	ISBN	YEAR_PUBLISHED	PRICE
1	Learn Java with Projects	Sean Kennedy and Maaike van Putten	Packt	978-1837637188	2023	39
2	Java Memory Management	Maaike van Putten and Sean Kennedy	Packt	978-1801812856	2022	34

(2 rows, 1 ms)

Edit

i. Retrieve all books.

Save

GET http://localhost:8080/books Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Query Params

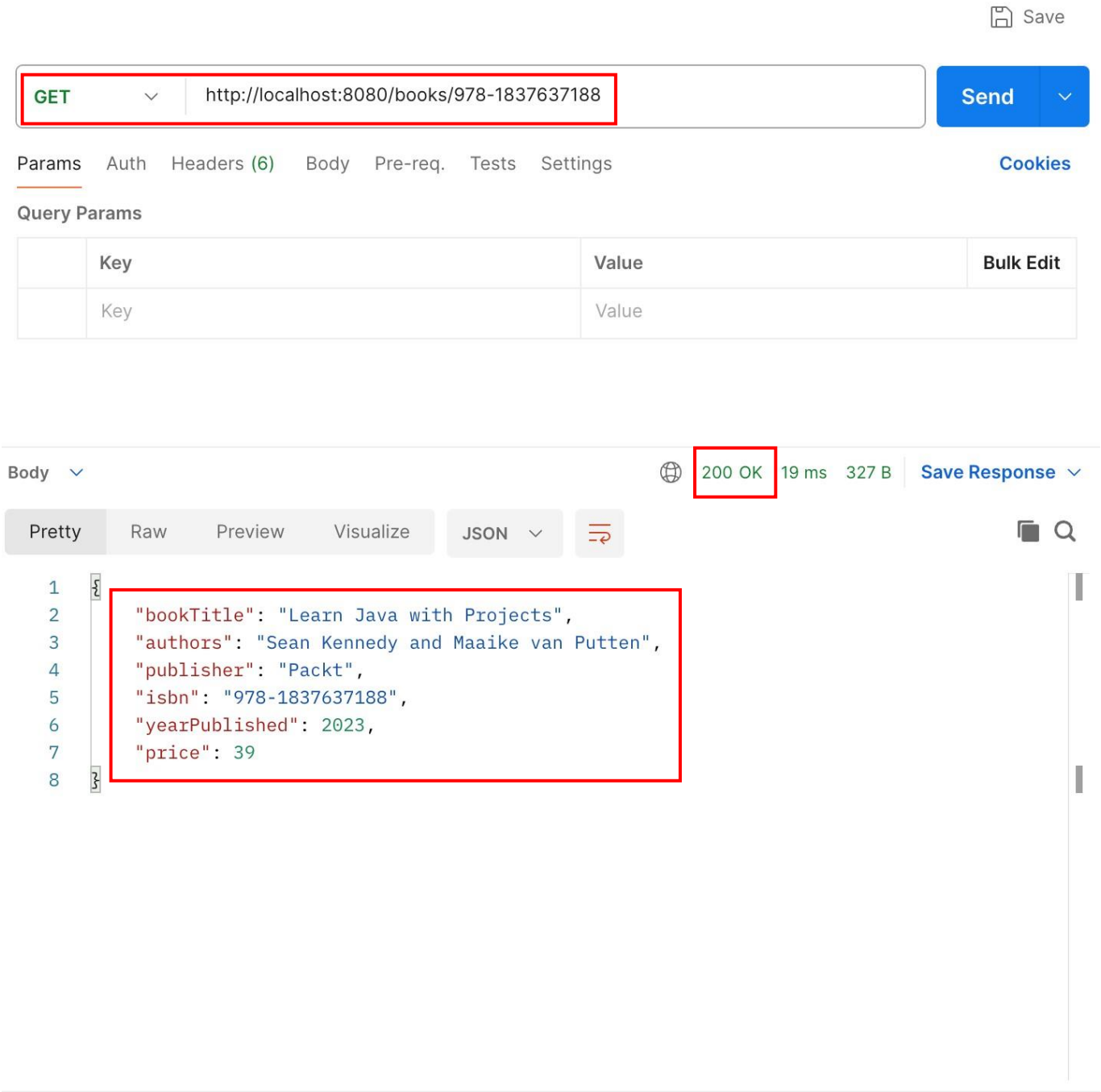
	Key	Value	Bulk Edit
	Key	Value	

Body 200 OK 25 ms 491 B Save Response

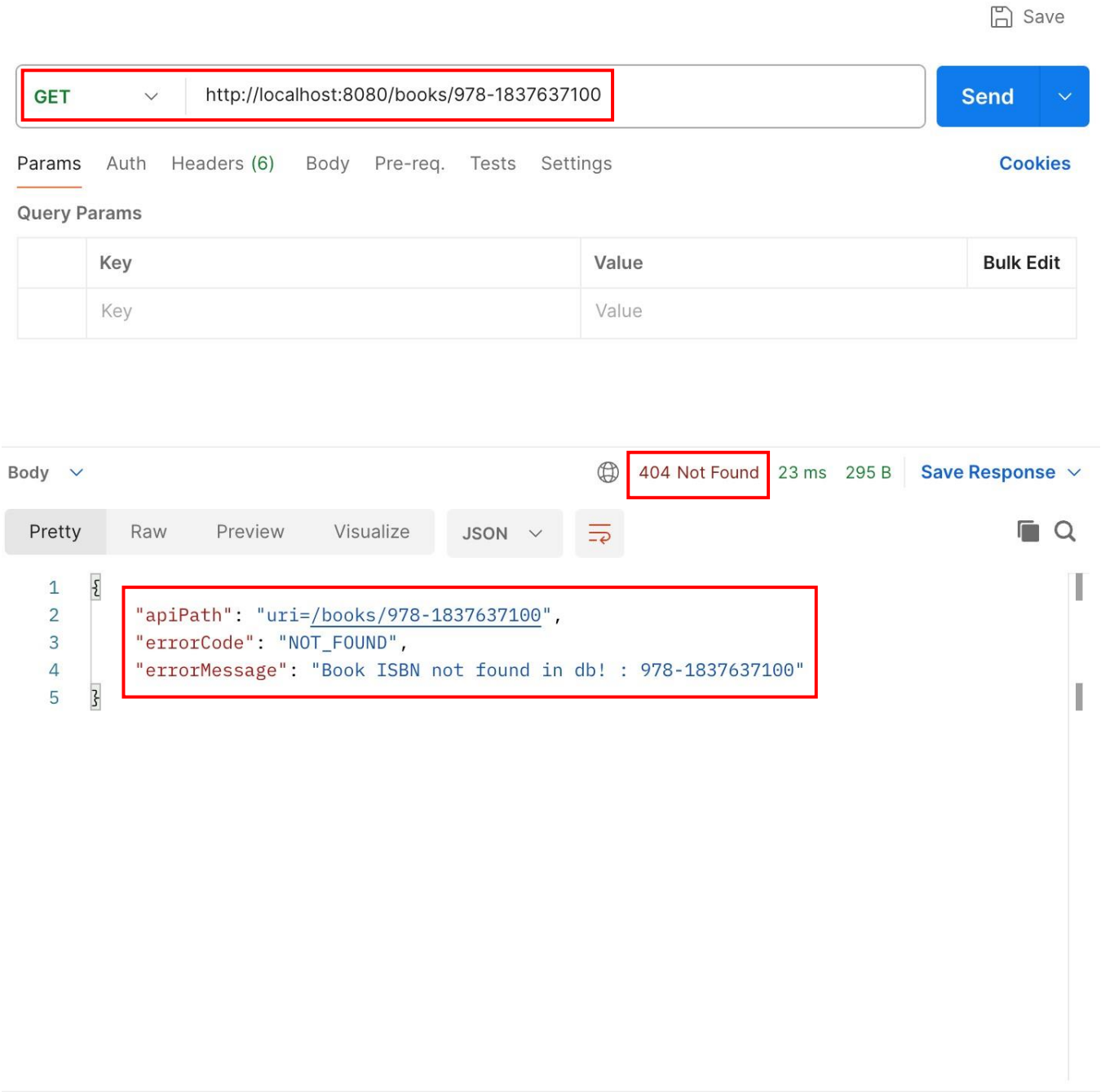
Pretty Raw Preview Visualize JSON


```
[
    {
        "bookTitle": "Learn Java with Projects",
        "authors": "Sean Kennedy and Maaike van Putten",
        "publisher": "Packt",
        "isbn": "978-1837637188",
        "yearPublished": 2023,
        "price": 39
    },
    {
        "bookTitle": "Java Memory Management",
        "authors": "Maaike van Putten and Sean Kennedy",
        "publisher": "Packt",
        "isbn": "978-1801812856",
        "yearPublished": 2022,
        "price": 34
    }
]
```

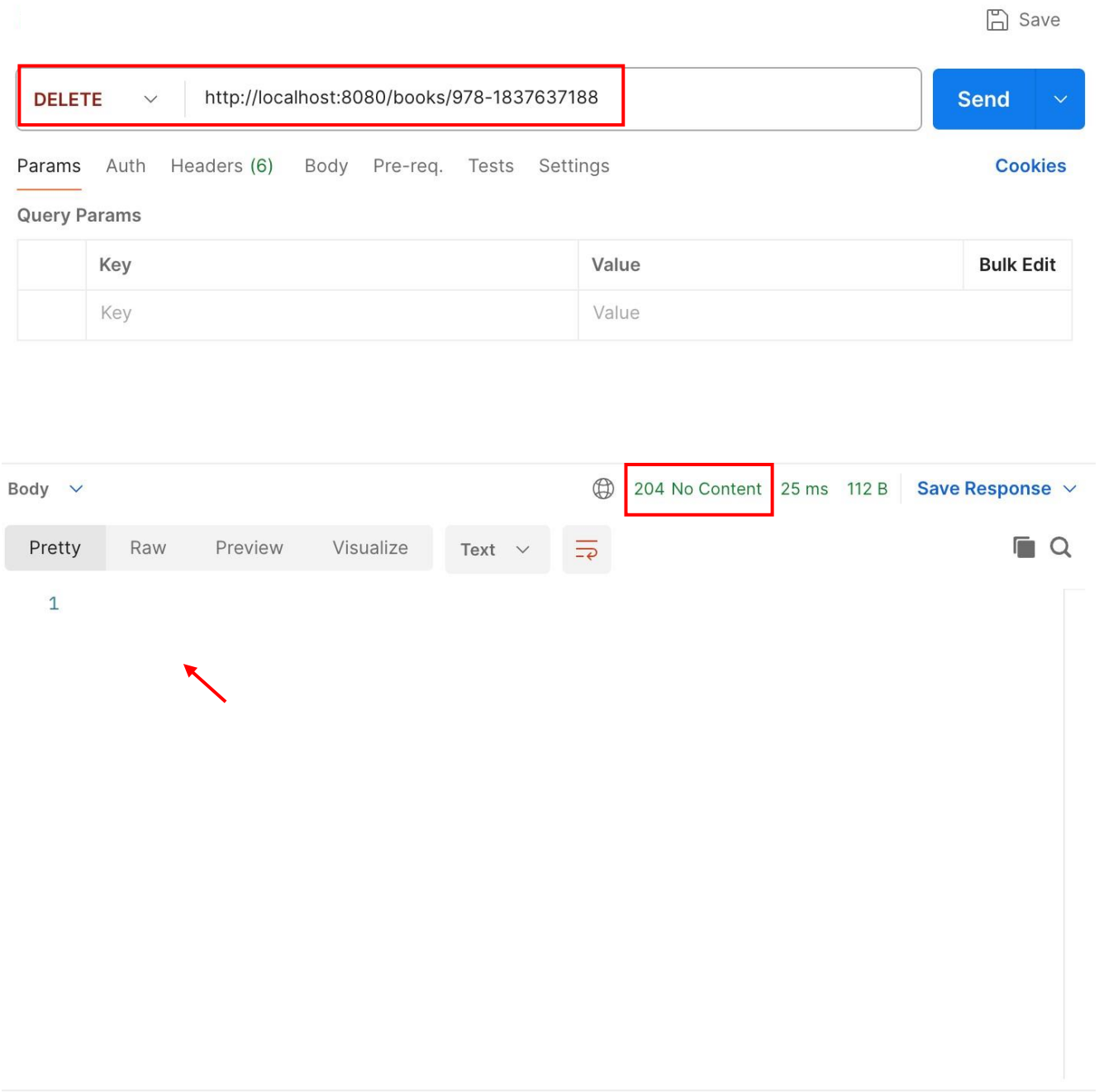
j. Retrieve one book by ISBN.



k. Retrieving a book with invalid ISBN.



l. Removing one book.



m. Database after one book removed.

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM BOOK
```

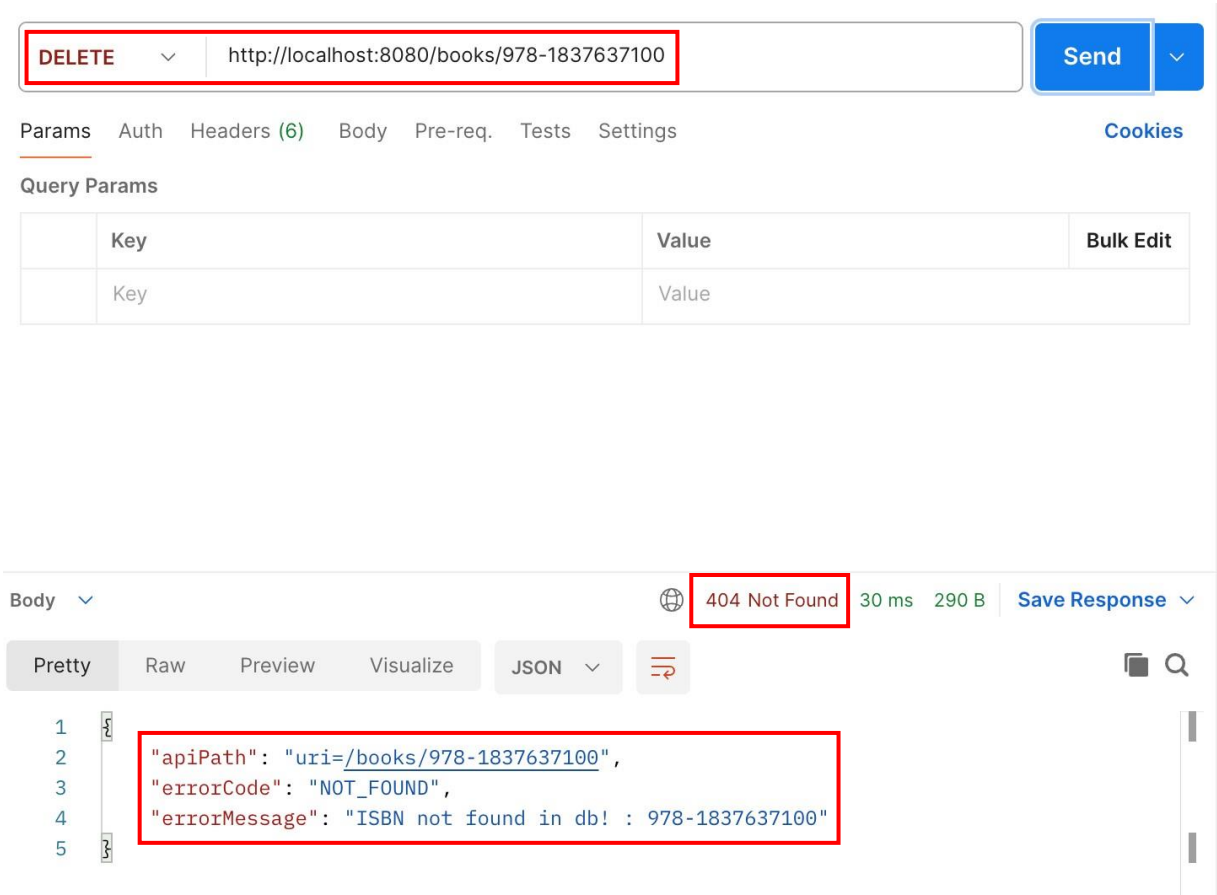
SELECT * FROM BOOK;

ID	BOOK_TITLE	AUTHORS	PUBLISHER	ISBN	YEAR_PUBLISHED	PRICE
2	Java Memory Management	Maaike van Putten and Sean Kennedy	Packt	978-1801812856	2022	34

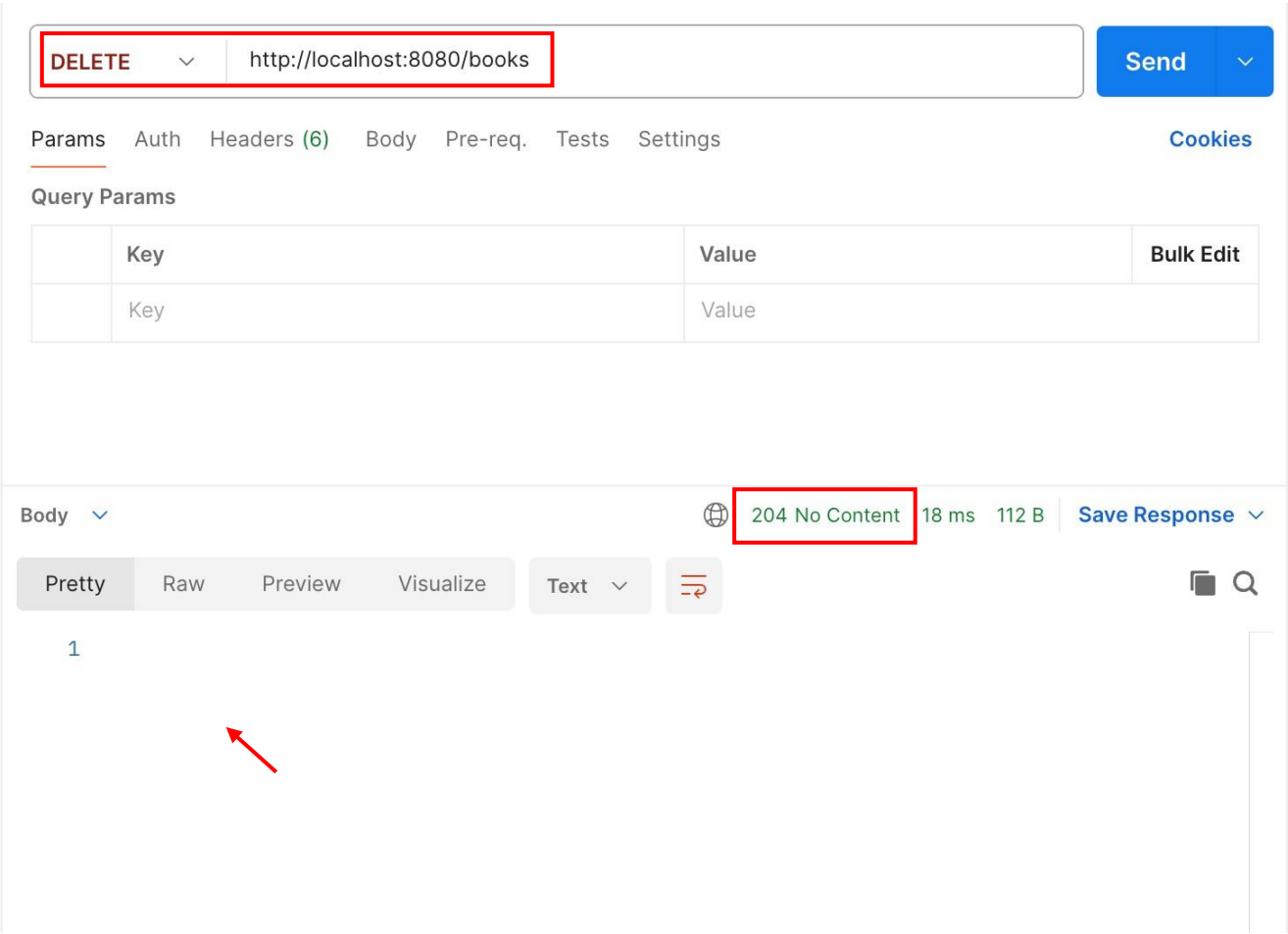
(1 row, 1 ms)

Edit

n. Removing a book with invalid ISBN.



o. Removing all books from database.



p. Database after all books removed.

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM BOOK
```

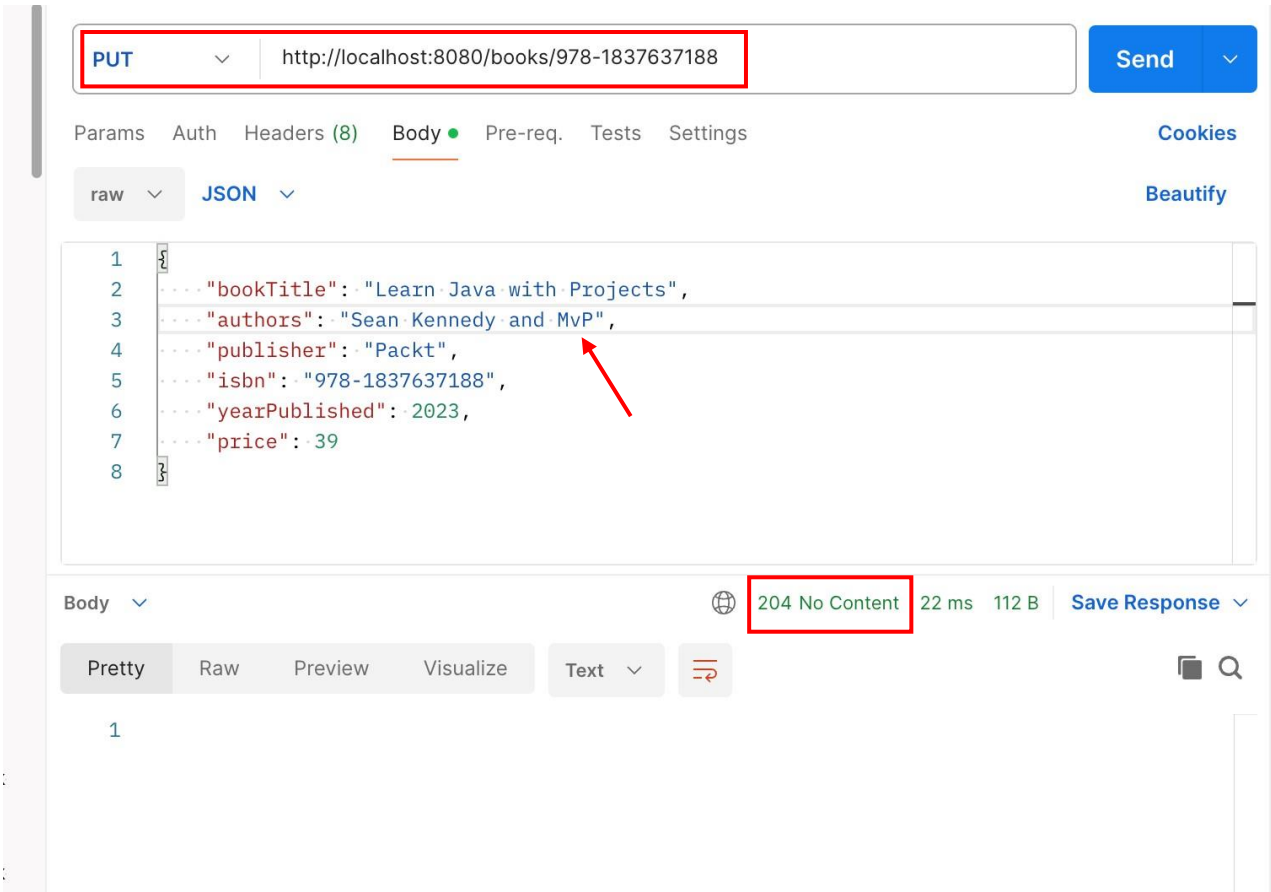
SELECT * FROM BOOK;

ID	BOOK_TITLE	AUTHORS	PUBLISHER	ISBN	YEAR_PUBLISHED	PRICE

(no rows, 1 ms)

Edit

q. Re-insert the same two books from the previous steps. Now update one of the books as follows: for the book with ISBN 978-1837637188, change “Maaike van Putten” to “MvP”.



r. Database after successful update.

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM BOOK
```

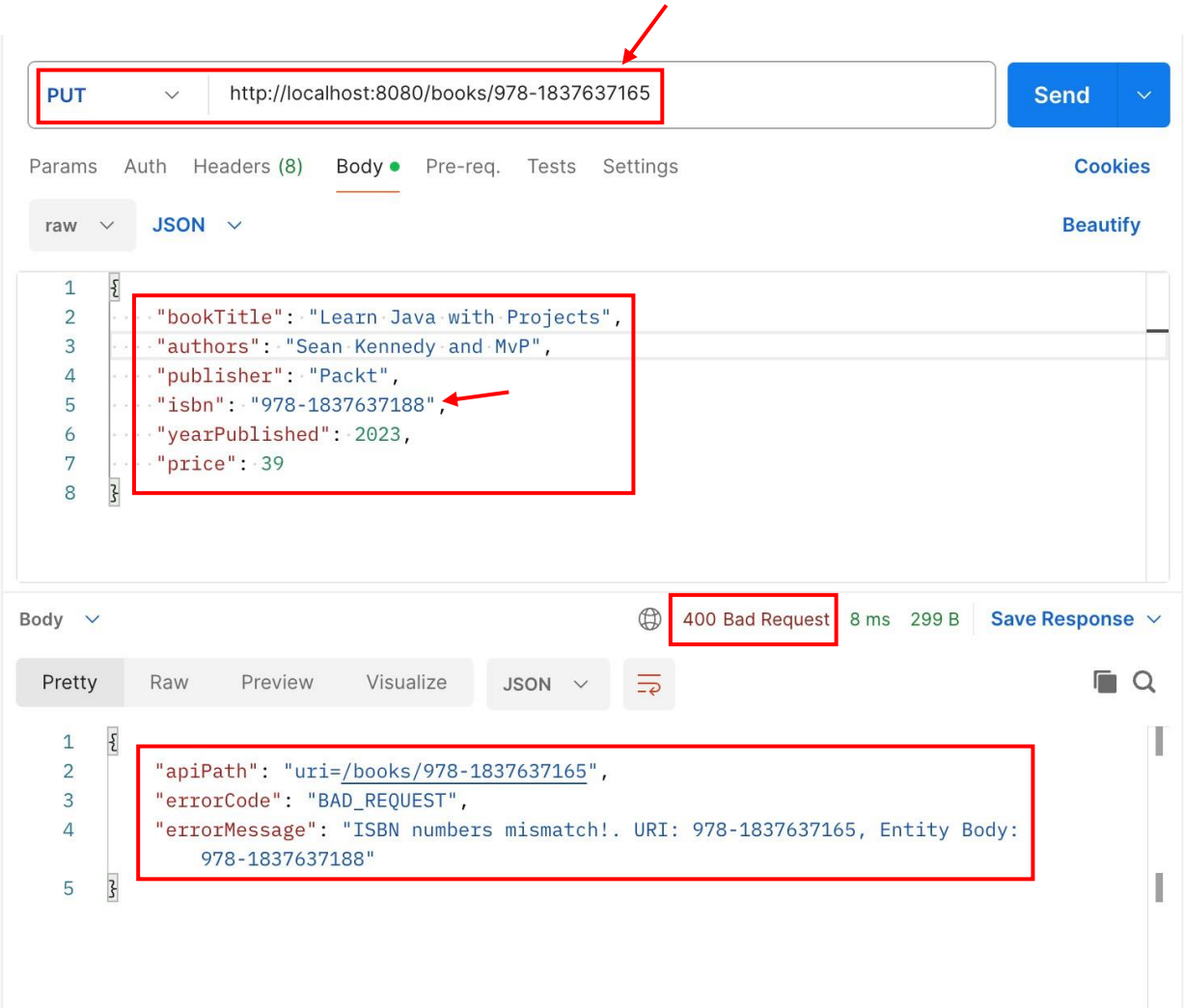
SELECT * FROM BOOK;

ID	BOOK_TITLE	AUTHORS	PUBLISHER	ISBN	YEAR_PUBLISHED	PRICE
3	Learn Java with Projects	Sean Kennedy and MvP	Packt	978-1837637188	2023	39
4	Java Memory Management	Maaike van Putten and Sean Kennedy	Packt	978-1801812856	2022	34

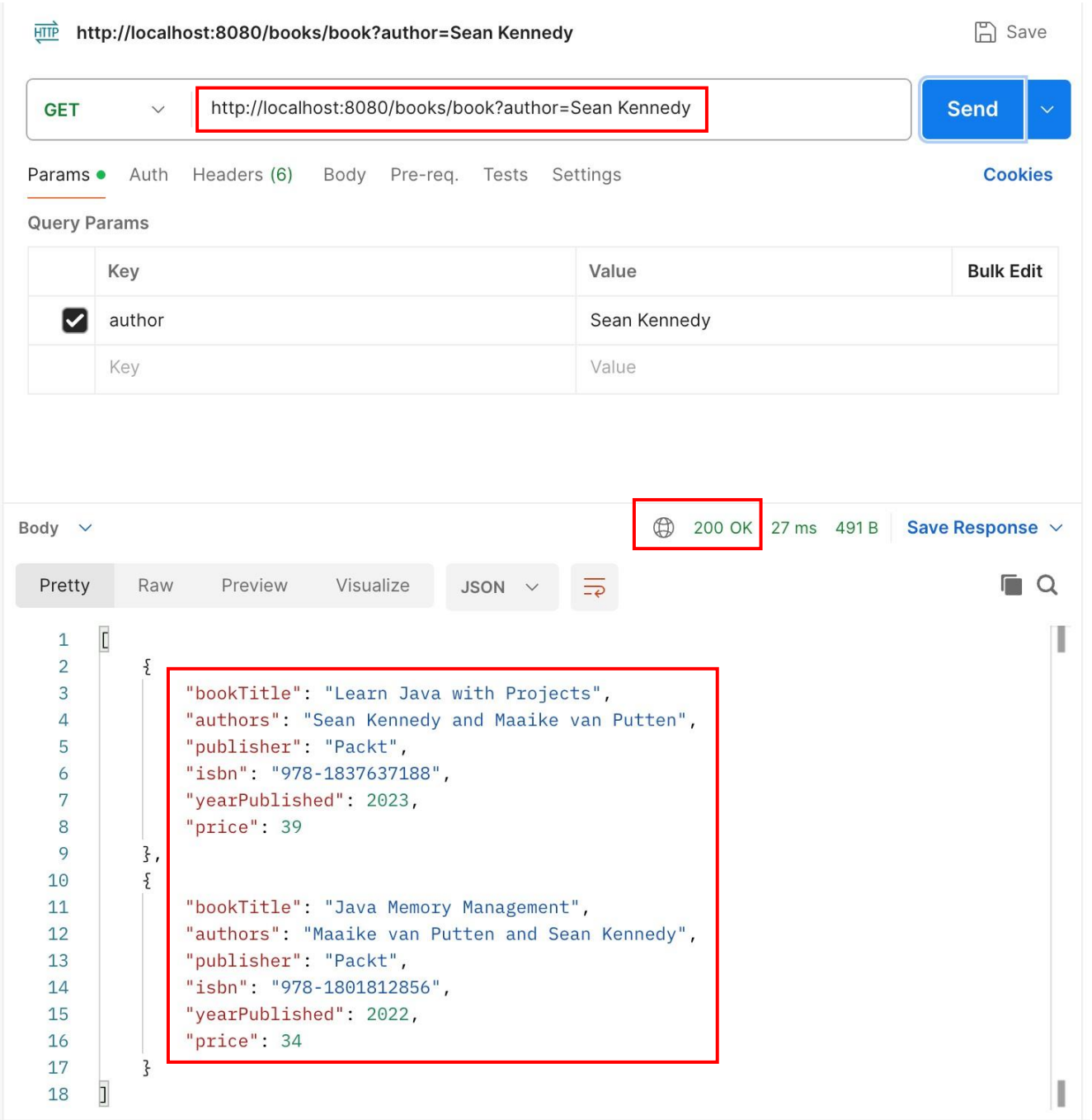
(2 rows, 0 ms)

Edit

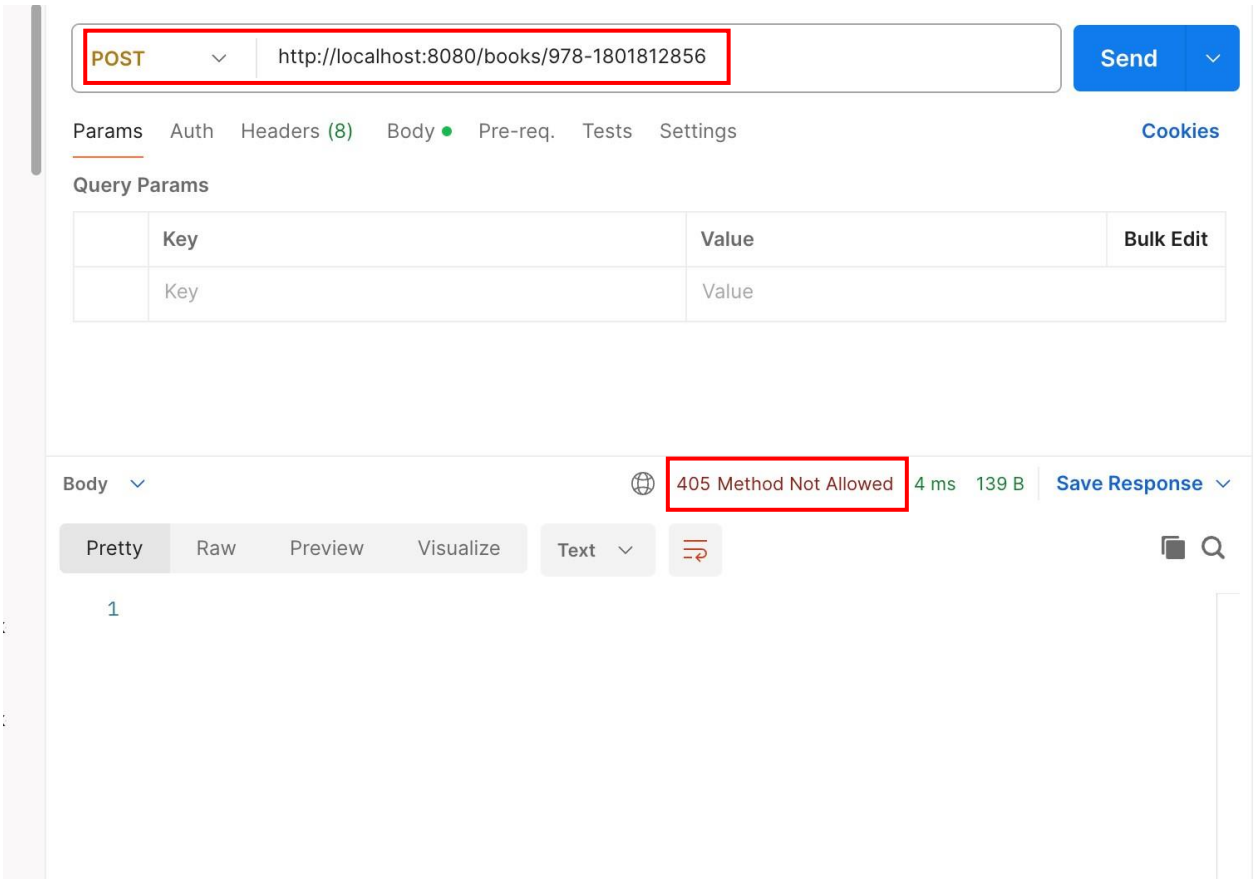
s. Attempting an update but there is a mismatch between the ISBNs in the URL and the message body.



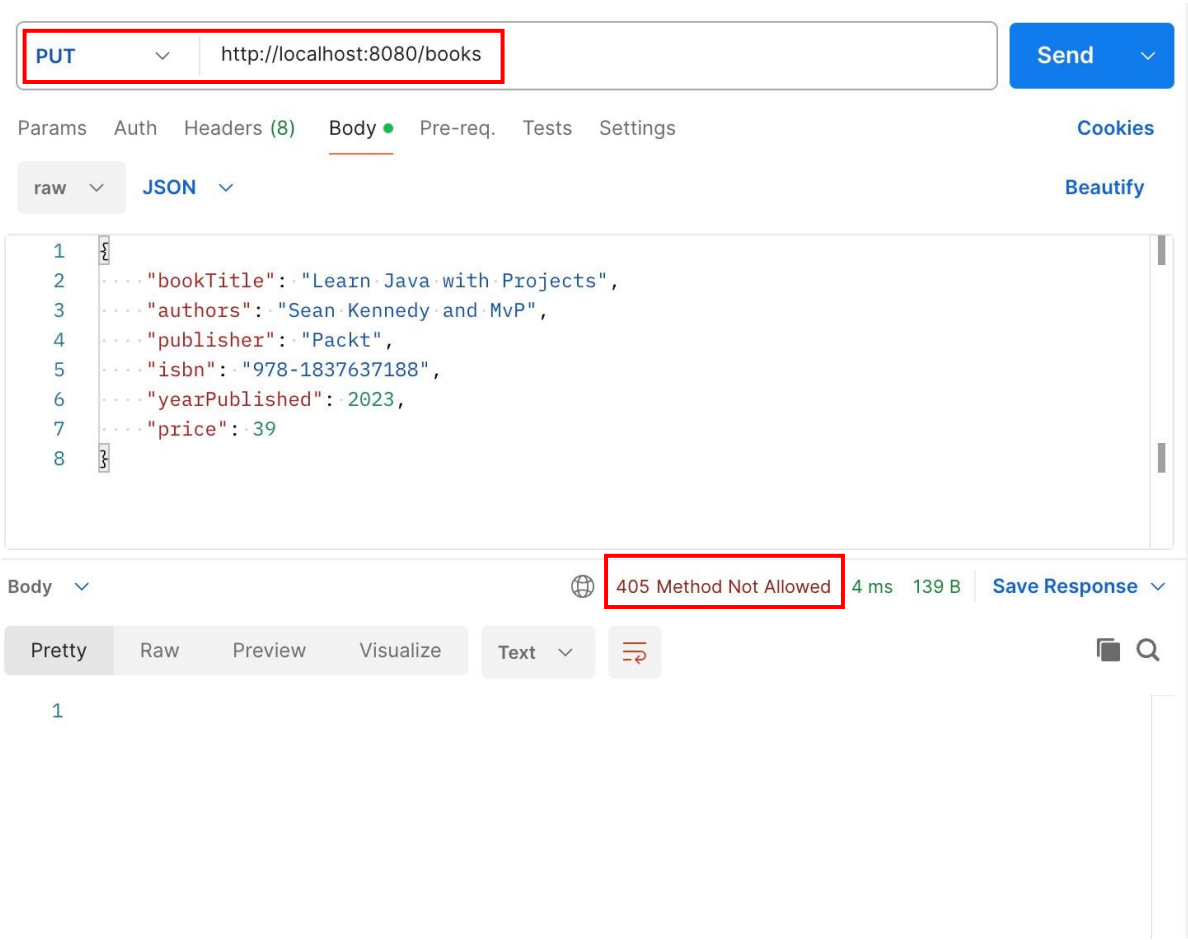
t. Using the query/request parameter “author”, retrieve all books authored by “Sean Kennedy”.



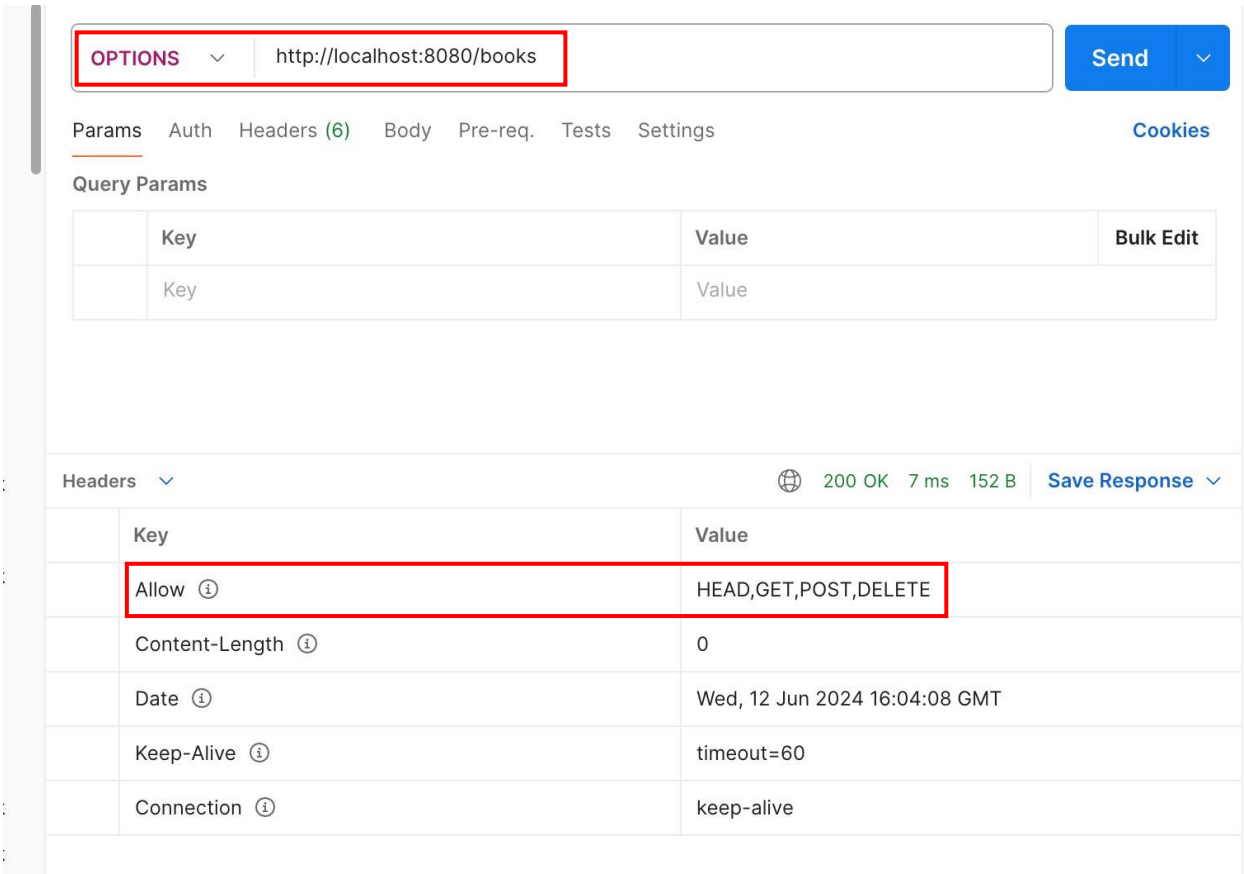
u. POST not allowed on an individual resource.



v. PUT not allowed on a collection.



w. OPTIONS on a collection



x. OPTIONS on an individual resource.