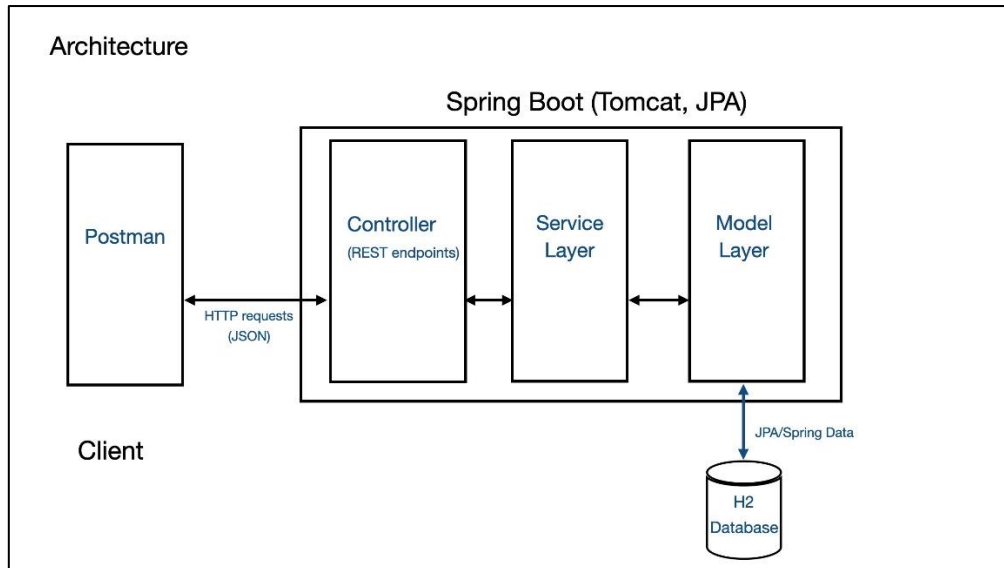


Spring Boot Assignment

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:

Spring Initializr

start.spring.io

Project

☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ Java ☐ Kotlin ☐ Groovy

☒ Maven

Spring Boot

☐ 3.3.1 (SNAPSHOT) ☒ 3.3.0 ☐ 3.2.7 (SNAPSHOT) ☐ 3.2.6

Project Metadata

Group: com.library

Artifact: books

Name: books

Description: Demo project for Spring Boot

Package name: com.library.books

Packaging: ☒ Jar ☐ War

Java: ☐ 22 ☐ 21 ☒ 17

Dependencies

Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

H2 Database SQL

Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

Spring Data JPA SQL

Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

Lombok DEVELOPER TOOLS

Java annotation library which helps to reduce boilerplate code.

GENERATE EXPLORE SHARE...

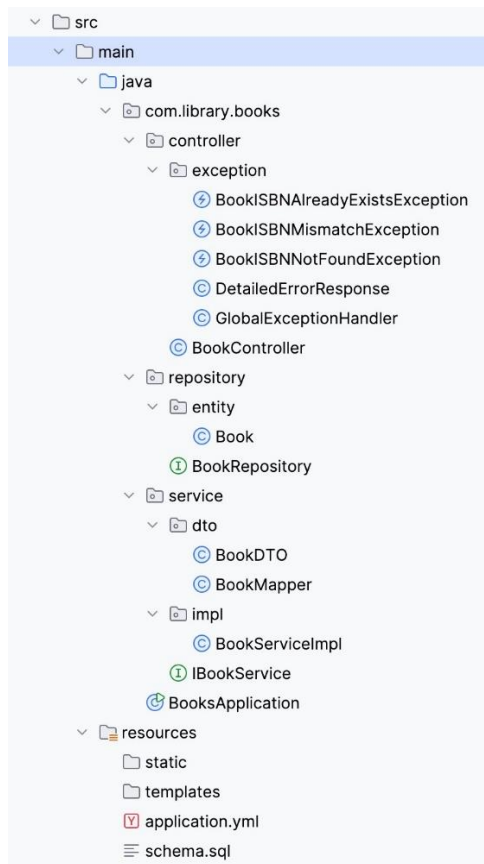
2. The *book* table in the database is created by the provided *schema.sql*.

```
1 CREATE TABLE IF NOT EXISTS `book` (  
2   `id` int AUTO_INCREMENT PRIMARY KEY,  
3   `book_title` varchar(100) NOT NULL,  
4   `authors` varchar(100) NOT NULL,  
5   `publisher` varchar(100) NOT NULL,  
6   `isbn` varchar(30) NOT NULL,  
7   `year_published` integer NOT NULL,  
8   `price` integer NOT NULL  
9 );
```

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

```
1 server:  
2   port: 8080  
3 spring:  
4   datasource:  
5     url: jdbc:h2:mem:testdb  
6     driverClassName: org.h2.Driver  
7     username: sa  
8     password: ''  
9   h2:  
10    console:  
11      enabled: true  
12  jpa:  
13    database-platform: org.hibernate.dialect.H2Dialect  
14    hibernate:  
15      ddl-auto: update  
16    show-sql: true
```

4. Packages overview:



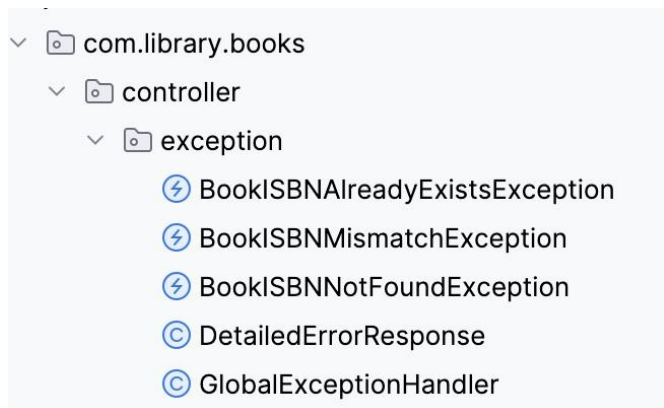
5. The URI/verb design. with associates semantics, is as follows:

- 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;`

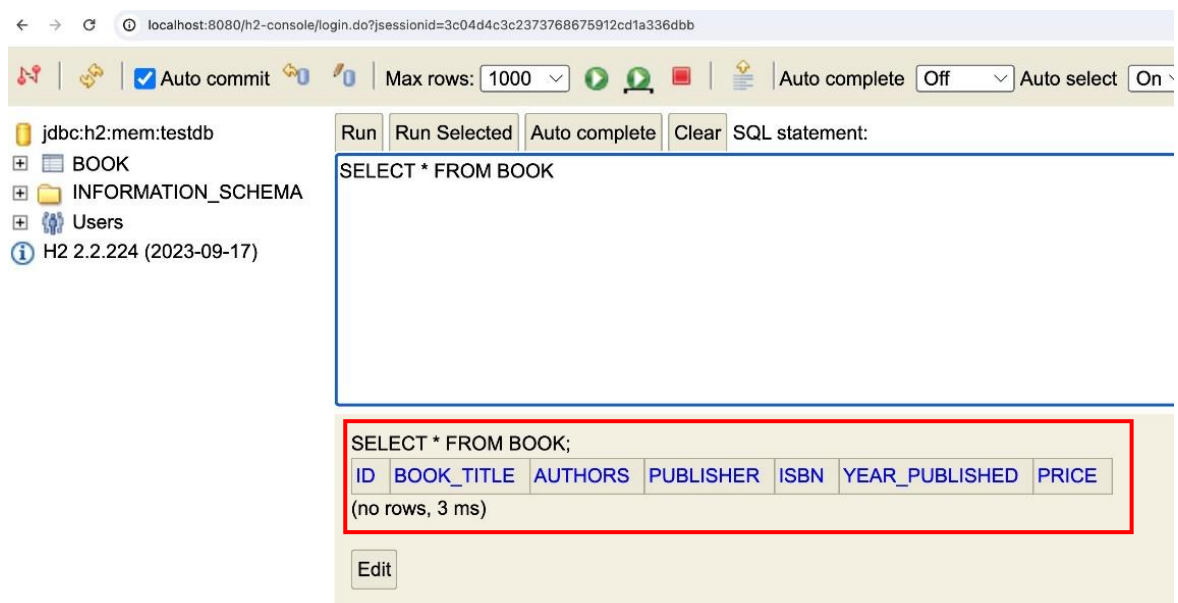
```
21 // SQL needs to be exact, would prefer a wildcard
22 @Query("select b from Book b where authors like %:authorName%") 1 usage
23 List<Book> findBooksByAuthorName(@Param("authorName") String authorName);
```

6. Exceptions - please cater for the following exceptions using a global exception controller:
- If you are trying to insert a book (POST) where the ISBN already exists in the database, generate a *BookISBNAlreadyExistsException*.
 - 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*.
 - 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).

The screenshot displays a REST client interface. At the top right, there is a 'Save' icon. The main interface is divided into two sections. The top section is for the request, showing a 'POST' method and the URL 'http://localhost:8080/books'. Below this, there are tabs for 'Params', 'Auth', 'Headers (8)', 'Body', 'Pre-req.', 'Tests', and 'Settings'. The 'Body' tab is selected, and the content is in 'JSON' format. The JSON body is:

```
{  "bookTitle": "Learn Java with Projects",  "authors": "Sean Kennedy and Maaïke van Putten",  "publisher": "Packt",  "isbn": "978-1837637188",  "yearPublished": 2023,  "price": 39}
```

 The bottom section shows the response, with a status of '201 Created', a time of '171 ms', and a size of '386 B'. The response body is also in 'JSON' format and is identical to the request body:

```
{  "bookTitle": "Learn Java with Projects",  "authors": "Sean Kennedy and Maaïke van Putten",  "publisher": "Packt",  "isbn": "978-1837637188",  "yearPublished": 2023,  "price": 39}
```

 A red arrow points to the 'Body' tab in the response section.

Save

POST http://localhost:8080/books Send

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

raw JSON

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

Body 201 Created 171 ms 386 B Save Response

Pretty Raw Preview Visualize JSON

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

c. Add first book (2 of 2).

Save

POST http://localhost:8080/books Send

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

raw JSON

```
1 {
2   "bookTitle": "Learn Java with Projects",
3   "authors": "Sean Kennedy and Maaïke 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.

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 Maaïke van Putten	Packt	978-1837637188	2023	39

(1 row, 1 ms)

Edit

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

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:8080/books
- Body Type:** JSON
- Request Body (JSON):**

```
{  "bookTitle": "Learn Java with Projects",  "authors": "Sean Kennedy and Maaïke van Putten",  "publisher": "Packt",  "isbn": "978-1837637188",  "yearPublished": 2023,  "price": 39}
```
- Response Status:** 400 Bad Request (highlighted in red)
- Response Time:** 10 ms
- Response Size:** 260 B
- Response Body (JSON):**

```
{  "apiPath": "uri=/books",  "errorCode": "BAD_REQUEST",  "errorMessage": "That book already exists in db! : 978-1837637188"}
```

f. Add second book (1 of 2).

The screenshot displays a REST client interface with a POST request to `http://localhost:8080/books`. The request body is a JSON object containing book details. The response status is 201 Created, and the response body is the same JSON object as the request body. Red boxes highlight the request URL, the request body, the response status, and the response body. A red arrow points to the 'Body' tab in the response section.

Save

POST `http://localhost:8080/books` Send

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

raw JSON

```
1 {
2   "bookTitle": "Java Memory Management",
3   "authors": "Maaïke van Putten and Sean Kennedy",
4   "publisher": "Packt",
5   "isbn": "978-1801812856",
6   "yearPublished": 2022,
7   "price": 34
8 }
```

Body 201 Created 13 ms 384 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "bookTitle": "Java Memory Management",
3   "authors": "Maaïke van Putten and Sean Kennedy",
4   "publisher": "Packt",
5   "isbn": "978-1801812856",
6   "yearPublished": 2022,
7   "price": 34
8 }
```

g. Add second book (2 of 2).

Save

POST http://localhost:8080/books Send

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

raw JSON

```
1 {
2   "bookTitle": "Java Memory Management",
3   "authors": "Maaïke van Putten and Sean Kennedy",
4   "publisher": "Packt",
5   "isbn": "978-1801812856",
6   "yearPublished": 2022,
7   "price": 34
8 }
9
```

Headers

201 Created 13 ms 384 B Save Response

Key	Value
Location ⓘ	http://localhost:8080/books/978-1801812856
Content-Type ⓘ	application/json
Transfer-Encoding ⓘ	chunked
Date ⓘ	Wed, 12 Jun 2024 15:39:44 GMT
Keep-Alive ⓘ	timeout=60
Connection ⓘ	keep-alive

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 ▼

ParamsAuthHeaders (6)BodyPre-req. TestsSettingsCookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	


Body ▼

 200 OK25 ms491 BSave Response ▼

PrettyRawPreviewVisualizeJSON ▼

```
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 {
10  "bookTitle": "Java Memory Management",
11  "authors": "Maaike van Putten and Sean Kennedy",
12  "publisher": "Packt",
13  "isbn": "978-1801812856",
14  "yearPublished": 2022,
15  "price": 34
16 }
17 }
18 }
```

j. Retrieve one book by ISBN.

 Save

GET

▼

http://localhost:8080/books/978-1837637188

Send ▼


ParamsAuthHeaders (6)BodyPre-req.TestsSettings


Cookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Body ▼


 200 OK19 ms327 BSave Response ▼

PrettyRawPreviewVisualizeJSON ▼

12345678

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

k. Retrieving a book with invalid ISBN.

 Save

GET

http://localhost:8080/books/978-1837637100

Send

ParamsAuthHeaders (6)BodyPre-req. TestsSettingsCookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Body

404 Not Found23 ms295 BSave Response

PrettyRawPreviewVisualizeJSON

```
1 {
2   "apiPath": "uri=/books/978-1837637100",
3   "errorCode": "NOT_FOUND",
4   "errorMessage": "Book ISBN not found in db! : 978-1837637100"
5 }
```

1. Removing one book.

Save

DELETE

http://localhost:8080/books/978-1837637188

Send

Params

Auth

Headers (6)

Body

Pre-req.

Tests

Settings

Cookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Body

204 No Content

25 ms

112 B

Save Response

Pretty

Raw

Preview

Visualize

Text

1

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.

DELETE

http://localhost:8080/books/978-1837637100

Send

ParamsAuthHeaders (6)BodyPre-req. TestsSettingsCookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Body

404 Not Found30 ms290 BSave Response

PrettyRawPreviewVisualizeJSON

```
1 {
2   "apiPath": "uri=/books/978-1837637100",
3   "errorCode": "NOT_FOUND",
4   "errorMessage": "ISBN not found in db! : 978-1837637100"
5 }
```

o. Removing all books from database.

DELETE

http://localhost:8080/books

Send

ParamsAuthHeaders (6)BodyPre-req. TestsSettingsCookies

Query Params


	Key	Value	Bulk Edit
	Key	Value	

Body

204 No Content18 ms112 BSave Response

PrettyRawPreviewVisualizeText

1



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 “Maaïke van Putten” to “MvP”.

The screenshot shows a REST client interface with a PUT request to `http://localhost:8080/books/978-1837637188`. The request body is a JSON object with the following fields: `bookTitle`, `authors`, `publisher`, `isbn`, `yearPublished`, and `price`. A red arrow points to the `authors` field, which contains the string `"Sean Kennedy and MvP"`. The response status is `204 No Content`, with a response time of 22 ms and a size of 112 B. The response body is empty.

```
PUT http://localhost:8080/books/978-1837637188
```

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

raw JSON

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

Body 204 No Content 22 ms 112 B Save Response

Pretty Raw Preview Visualize Text

1

r. Database after successful update.

RunRun SelectedAuto completeClear

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.

The screenshot shows a REST client interface with a PUT request to `http://localhost:8080/books/978-1837637165`. The request body is a JSON object with the following fields:

```
{
  "bookTitle": "Learn Java with Projects",
  "authors": "Sean Kennedy and MVP",
  "publisher": "Packt",
  "isbn": "978-1837637188",
  "yearPublished": 2023,
  "price": 39
}
```

The response is a 400 Bad Request with the following JSON body:

```
{
  "apiPath": "uri=/books/978-1837637165",
  "errorCode": "BAD_REQUEST",
  "errorMessage": "ISBN numbers mismatch!. URI: 978-1837637165, Entity Body: 978-1837637188"
}
```

Red boxes and arrows highlight the mismatch between the ISBN in the URL (978-1837637165) and the ISBN in the request body (978-1837637188), and the corresponding error message in the response.

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

HTTP <http://localhost:8080/books/book?author=Sean Kennedy> Save

GET ▼ <http://localhost:8080/books/book?author=Sean Kennedy> Send ▼

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

Query Params

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	author	Sean Kennedy	
	Key	Value	

Body ▼ 200 OK 27 ms 491 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼ ≡

```
1  [
2    {
3      "bookTitle": "Learn Java with Projects",
4      "authors": "Sean Kennedy and Maaïke van Putten",
5      "publisher": "Packt",
6      "isbn": "978-1837637188",
7      "yearPublished": 2023,
8      "price": 39
9    },
10   {
11     "bookTitle": "Java Memory Management",
12     "authors": "Maaïke van Putten and Sean Kennedy",
13     "publisher": "Packt",
14     "isbn": "978-1801812856",
15     "yearPublished": 2022,
16     "price": 34
17   }
18 ]
```


u. POST not allowed on an individual resource.

The screenshot shows a REST client interface with a red box highlighting the **POST** method and the URL `http://localhost:8080/books/978-1801812856`. Below the URL bar, the **Params** tab is selected, showing a table for **Query Params** with columns **Key**, **Value**, and **Bulk Edit**. The **Body** tab is also visible. The response section shows a **405 Method Not Allowed** status, with a response time of **4 ms** and a size of **139 B**. The response body is displayed in **Text** format, showing the number **1**.

POST ⌵ `http://localhost:8080/books/978-1801812856` Send ⌵

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

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Body ⌵ ⌕ **405 Method Not Allowed** 4 ms 139 B Save Response ⌵

Pretty Raw Preview Visualize Text ⌵ 📄 🔍

1

v. PUT not allowed on a collection.

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** http://localhost:8080/books
- Body:** JSON
- JSON Body:**

```
{
  "bookTitle": "Learn Java with Projects",
  "authors": "Sean Kennedy and MVP",
  "publisher": "Packt",
  "isbn": "978-1837637188",
  "yearPublished": 2023,
  "price": 39
}
```
- Response:** 405 Method Not Allowed (4 ms, 139 B)
- Response Body:** Pretty

w. OPTIONS on a collection

OPTIONS

http://localhost:8080/books

Send

Params

Auth

Headers (6)

Body

Pre-req.

Tests

Settings

Cookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Headers

200 OK 7 ms 152 B

Save Response

	Key	Value
	Allow ⓘ	HEAD,GET,POST,DELETE
	Content-Length ⓘ	0
	Date ⓘ	Wed, 12 Jun 2024 16:04:08 GMT
	Keep-Alive ⓘ	timeout=60
	Connection ⓘ	keep-alive

x. OPTIONS on an individual resource.

OPTIONS

http://localhost:8080/books/978-1837637188

Send

Params

Auth

Headers (6)

Body

Pre-req.

Tests

Settings

Cookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Headers

200 OK 5 ms 151 B Save Response

	Key	Value
	Allow ⓘ	HEAD,GET,PUT,DELETE
	Content-Length ⓘ	0
	Date ⓘ	Fri, 14 Jun 2024 10:49:56 GMT
	Keep-Alive ⓘ	timeout=60
	Connection ⓘ	keep-alive