

Bookstore App Requirements

Table of Contents

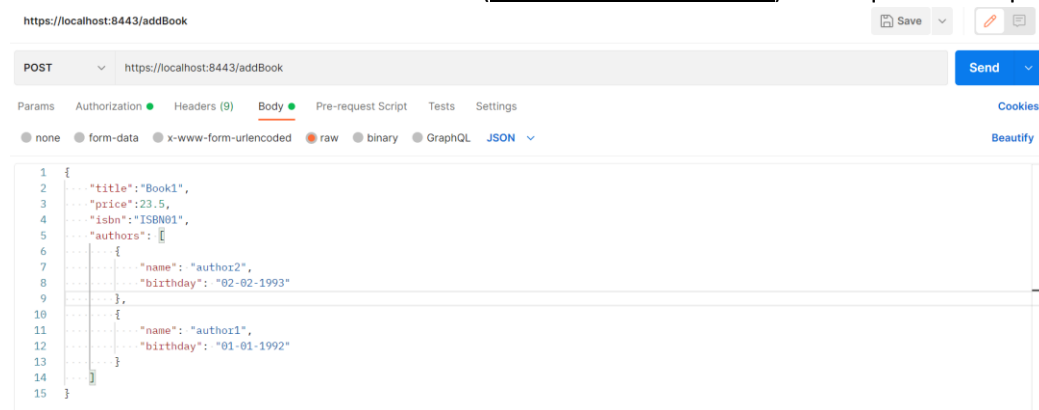
1.0 Book Controller	1
1.1 Add Book (/addBook).....	1
1.2 Return Books By Title (/ReturnBookByTitle).....	3
1.3 Return All Books (/ReturnAllBooks)	4
1.4 Return Book by Id (/ReturnBookById/{id})	5
1.5 Update Book By ISBN (/UpdateBookByIsbn/{isbn})	6
1.6 Delete Book by Id (/DeleteBookById/{id})	10
1.7 Return Books by Author's Name (/ReturnBooksByAuthorName/{author_name})	11
2.0 Author Controller	12
2.1 Add Author(/addAuthor)	12
2.2 Return Author by Id /ReturnAuthorById/{id})	13
Notes.....	15

1.0 Book Controller

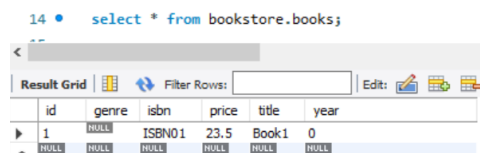
1.1 Add Book (/addBook)

▪ Add Book: Basic Functionality

User should be able to add a new book (with or without authors) but requires a unique ISBN.



Initial Database State



```
16 • select * from bookstore.author_book;
```

	author_id	book_id
▶	1	1
	2	1

```
18 • select * from bookstore.authors;
```

	id	birthday	name
▶	1	01-01-1992	author1
	2	02-02-1993	author2

▪ Add Book: Possible Scenario

The following are the results if users were to add books with different ISBN but with existing authors (i.e. same authors as another book) or a new author.

https://localhost:8443/addBook

POST https://localhost:8443/addBook

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "title": "Book2",
3   "price": 24.5,
4   "isbn": "ISBN02",
5   "authors": [
6     {
7       "name": "author1",
8       "birthday": "02-02-1993"
9     },
10    {
11      "name": "author2",
12      "birthday": "01-01-1992"
13    }
14  ]
15 }

```

```
14 • select * from bookstore.books;
```

	id	genre	isbn	price	title	year
▶	1	NULL	ISBN01	23.5	Book1	0
	2	NULL	ISBN02	24.5	Book2	0

```
16 • select * from bookstore.author_book;
```

	author_id	book_id
▶	1	1
	2	1
	1	2
	2	2

```
18 • select * from bookstore.authors;
```

	id	birthday	name
▶	1	01-01-1992	author2
	2	02-02-1993	author1

▪ Add Book: Error Handling

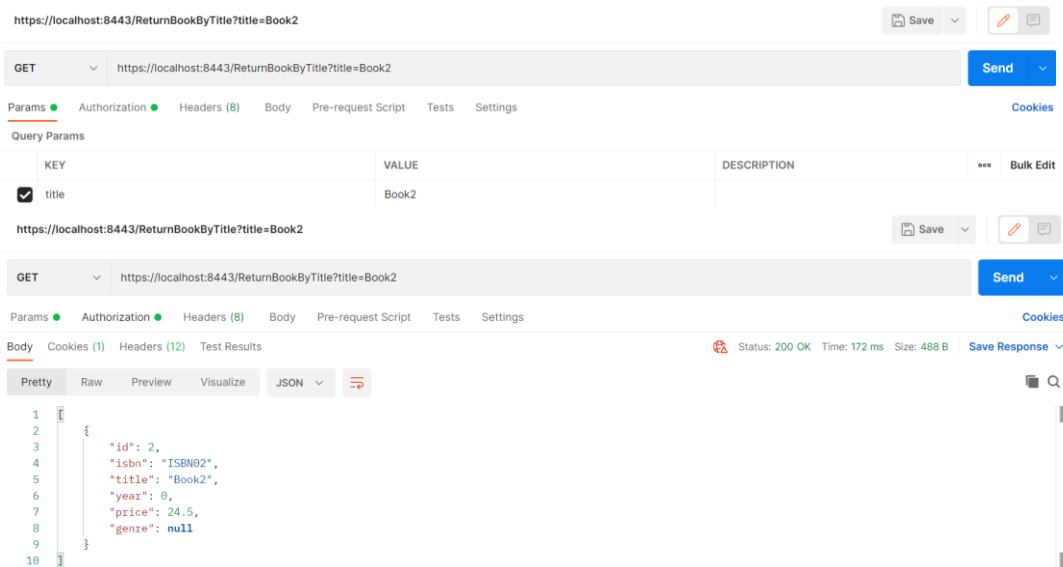
This is the error handling when users were to add duplicated Books of the same ISBN (unique constraint).



1.2 Return Books By Title (/ReturnBookByTitle)

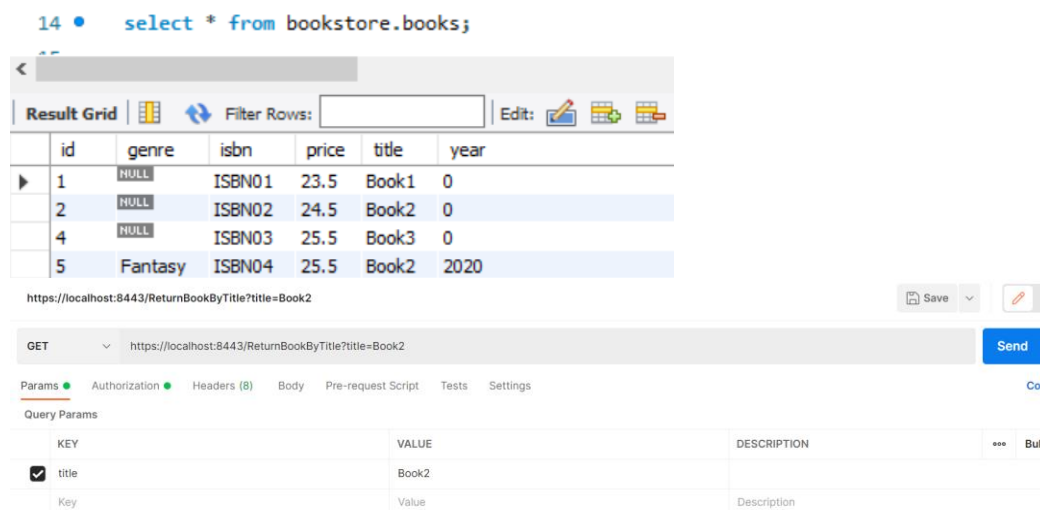
- Return Books By Title: Basic Functionality

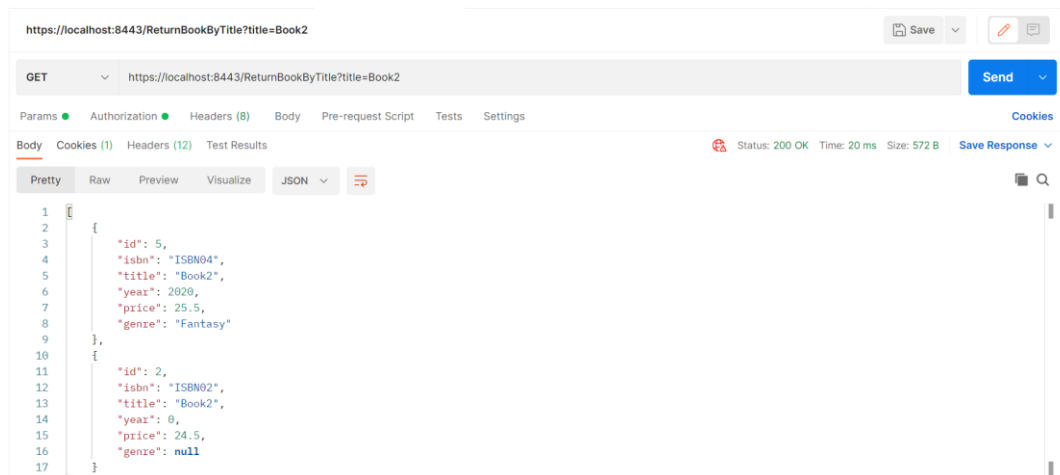
User should be able to search and retrieve books by passing Title as a query parameter.



- Return Book By Title: Scenarios

Users should also be able retrieve multiple books with the same titles (but different ISBN).





- **Return Book By Title: Error Handling**

There will be no content returned when book title does not exist.

14 • `select * from bookstore.books;`

< 15

Result Grid Filter Rows: Edit:

	id	genre	isbn	price	title	year
▶	1	NULL	ISBN01	23.5	Book1	0
	2	NULL	ISBN02	24.5	Book2	0
	4	NULL	ISBN03	25.5	Book3	0
	5	Fantasy	ISBN04	25.5	Book2	2020

https://localhost:8443/ReturnBookByTitle?title=Book4

GET https://localhost:8443/ReturnBookByTitle?title=Book4 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
<input checked="" type="checkbox"/> title	Book4		
Key	Value	Description	

https://localhost:8443/ReturnBookByTitle?title=Book4

GET https://localhost:8443/ReturnBookByTitle?title=Book4 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

Body Cookies (1) Headers (10) Test Results Status: 204 No Content Time: 24 ms Size: 359 B Save Response

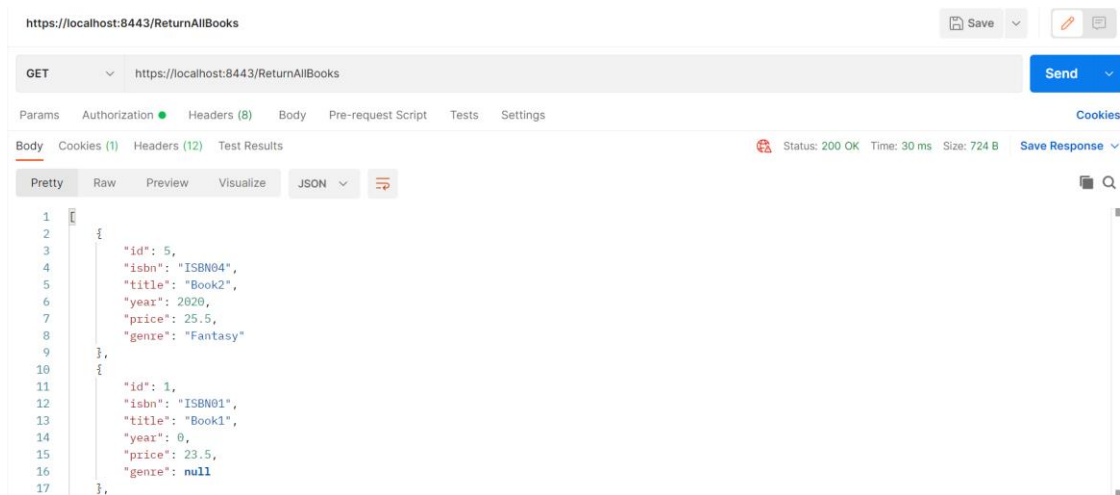
Pretty Raw Preview Visualize Text

1

1.3 Return All Books (/ReturnAllBooks)

- **Return All Books: Basic Functionality**

User should be able to query to get information on all existing books in the database. If there are no books in the database, no content will be returned.



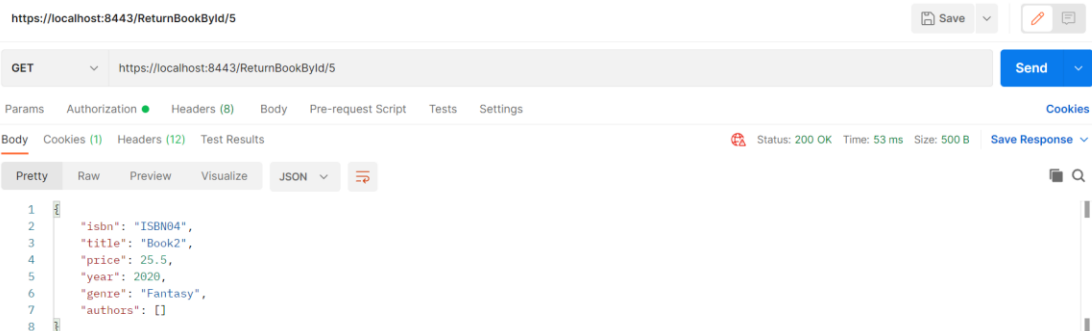
1.4 Return Book by Id (/ReturnBookById/{id})

- Return Book by Id: Basic Functionality**

User should be able to query to get information on a book through providing a Book ID.

```
14 • select * from bookstore.books;
```

	id	genre	isbn	price	title	year
▶	1	NULL	ISBN01	23.5	Book1	0
	2	NULL	ISBN02	24.5	Book2	0
	4	NULL	ISBN03	25.5	Book3	0
	5	Fantasy	ISBN04	25.5	Book2	2020

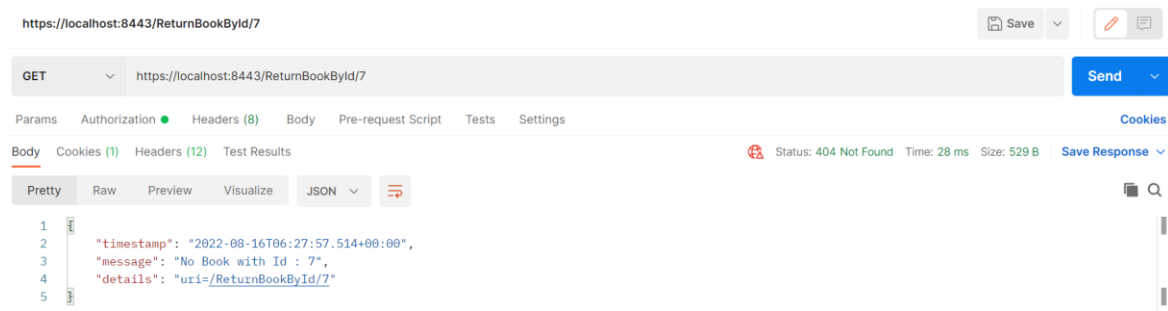


- Return Book by Id: Error Handling**

There will be no content returned when the book id does not exist.

```
14 • select * from bookstore.books;
```

	id	genre	isbn	price	title	year
▶	1	NULL	ISBN01	23.5	Book1	0
	2	NULL	ISBN02	24.5	Book2	0
	4	NULL	ISBN03	25.5	Book3	0
	5	Fantasy	ISBN04	25.5	Book2	2020



1.5 Update Book By ISBN (/UpdateBookByIsbn/{isbn})

- **Update Book by ISBN: Basic Functionality**

User should be able to update the basic properties, as well as the author list, of the book fields by passing in the ISBN as a parameter. ISBN is used as it is a unique identifier that is known to the user.

Initial Database State

14 • select * from bookstore.books;

Result Grid

	id	genre	isbn	price	title	year
▶	1	NULL	ISBN01	23.5	Book1	0
	2	NULL	ISBN02	24.5	Book2	0
	4	NULL	ISBN03	25.5	Book3	0
	5	Fantasy	ISBN04	25.5	Book2	2020

16 • select * from bookstore.author_book;

Result Grid

	author_id	book_id
▶	1	1
	2	1
	1	2
	2	2

18 • select * from bookstore.authors;

Result Grid

	id	birthday	name
▶	1	01-01-1992	author2
	2	02-02-1993	author1

- **Update Book by ISBN: Scenarios**

User can edit the basic properties (ISBN, genre, price, title, year) of a book, without updating the author list.

https://localhost:8443/UpdateBookByIsbn/ISBN02

PUT https://localhost:8443/UpdateBookByIsbn/ISBN02

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "title": "Book5",
3   "isbn": "ISBN05",
4   "price": 25.5,
5   "genre": "Fantasy",
6   "year": 2015,
7   "authors": [
8     {
9       "name": "author1",
10      "birthday": "02-02-1993"
11     },
12     {
13       "name": "author2",
14       "birthday": "01-01-1992"
15     }
16   ]
17 }
```

https://localhost:8443/UpdateBookByIsbn/ISBN02

PUT https://localhost:8443/UpdateBookByIsbn/ISBN02

Params Authorization Headers (10) Body Pre-request Script Tests Settings

Body Cookies (1) Headers (12) Test Results

Status: 200 OK Time: 39 ms Size: 456 B Save Response

Pretty Raw Preview Visualize Text

1 Book with ISBN : ISBN02 updated with success!

14 • select * from bookstore.books;

Result Grid Filter Rows: Edit:

	id	genre	isbn	price	title	year
1	1	NULL	ISBN01	23.5	Book1	0
2	2	Fantasy	ISBN05	25.5	Book5	2015
4	4	NULL	ISBN03	25.5	Book3	0
5	5	Fantasy	ISBN04	25.5	Book2	2020

16 • select * from bookstore.author_book;

Result Grid Filter Rows: Edit:

	author_id	book_id
1	1	1
2	2	1
1	1	2
2	2	2

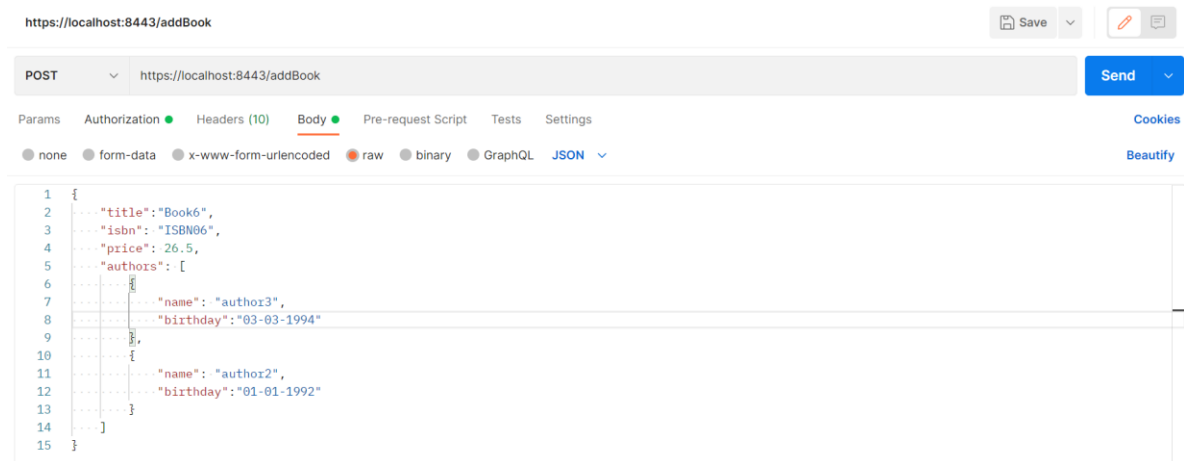
18 • select * from bookstore.authors;

Result Grid Filter Rows: Edit:

	id	birthday	name
1	1	01-01-1992	author2
2	2	02-02-1993	author1

User can edit the author list of the book by (a) adding or removing existing author, (b) adding a new author.

Initial Database State



14 • select * from bookstore.books;

Result Grid | Filter Rows: | Edit:

	id	genre	isbn	price	title	year
1	1	NULL	ISBN01	25.5	Book1	0
2	2	Fantasy	ISBN05	25.5	Book5	2015
4	4	NULL	ISBN03	25.5	Book3	0
5	5	Fantasy	ISBN04	25.5	Book2	2020
6	6	NULL	ISBN06	26.5	Book6	0

16 • select * from bookstore.author_book;

Result Grid | Filter Rows: | Edit:

	author_id	book_id
1	1	1
2	2	1
1	1	2
2	2	2
1	1	6
3	3	6

18 • select * from bookstore.authors;

Result Grid | Filter Rows: | Edit:

	id	birthday	name
1	1	01-01-1992	author2
2	2	02-02-1993	author1
3	3	03-03-1994	author3

Results

The pictures below show the initial state of Book5. We assume that user wish to remove author1 to replace with existing author3, then add a new author author4.

https://localhost:8443/UpdateBookByIsbn/ISBN02

Save



PUT https://localhost:8443/UpdateBookByIsbn/ISBN02

Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings

Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Beautify

```
1 {
2   "title": "Book5",
3   "isbn": "ISBN05",
4   "price": 25.5,
5   "genre": "Fantasy",
6   "year": 2015,
7   "authors": [
8     {
9       "name": "author1",
10      "birthday": "02-02-1993"
11    },
12    {
13      "name": "author2",
14      "birthday": "01-01-1992"
15    }
16  ]
17 }
```

https://localhost:8443/UpdateBookByIsbn/ISBN05

Save



PUT https://localhost:8443/UpdateBookByIsbn/ISBN05

Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings

Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Beautify

```
1 {
2   "title": "Book5",
3   "isbn": "ISBN05",
4   "price": 28,
5   "authors": [
6     {
7       "name": "author3",
8       "birthday": "03-03-1994"
9     },
10    {
11      "name": "author2",
12      "birthday": "01-01-1992"
13    },
14    {
15      "name": "author4",
16      "birthday": "04-04-1993"
17    }
18  ]
19 }
```

14 • select * from bookstore.books;

Result Grid Filter Rows: Edit

id	genre	isbn	price	title	year
1	NULL	ISBN01	25.5	Book1	0
2	NULL	ISBN05	28	Book5	0
4	NULL	ISBN03	25.5	Book3	0
5	Fantasy	ISBN04	25.5	Book2	2020
6	NULL	ISBN06	26.5	Book6	0

16 • select * from bookstore.author_book;

Result Grid Filter Rows: Edit

author_id	book_id
1	1
2	1
1	2
3	2
4	2
1	6
3	6

18 • select * from bookstore.authors;

Result Grid Filter Rows: Edit

id	birthday	name
1	01-01-1992	author2
2	02-02-1993	author1
3	03-03-1994	author3
4	04-04-1993	author4

1.6 Delete Book by Id (/DeleteBookById/{id})

- **Delete Book by Id: Basic Functionality**

User should be able to delete a book by passing the Book Id as a parameter. When a book is deleted, the associated author-book mapping should be deleted as well.

Initial Database State

```
14 • select * from bookstore.books;
```

id	genre	isbn	price	title	year
1	NULL	ISBN01	25.5	Book1	0
2	NULL	ISBN05	28	Book5	0
4	NULL	ISBN03	25.5	Book3	0
5	Fantasy	ISBN04	25.5	Book2	2020
6	NULL	ISBN06	26.5	Book6	0

```
16 • select * from bookstore.author_book;
```

author_id	book_id
1	1
2	1
1	2
3	2
4	2
1	6
3	6

```
18 • select * from bookstore.authors;
```

id	birthday	name
1	01-01-1992	author2
2	02-02-1993	author1
3	03-03-1994	author3
4	04-04-1993	author4

Results

The pictures below show the result when the user deletes the book with Book Id 2.

https://localhost:8443/DeleteBookById/2

DELETE https://localhost:8443/DeleteBookById/2

Params Authorization Headers (10) Body Pre-request Script Tests Settings

Body Cookies (1) Headers (12) Test Results

Status: 200 OK Time: 533 ms Size: 449 B Save Response

Pretty Raw Preview Visualize Text

1 Book with ID : 2 deleted with success!

14 • `select * from bookstore.books;`

Result Grid					
Filter Rows:					
id	genre	isbn	price	title	year
1	NULL	ISBN01	25.5	Book1	0
4	NULL	ISBN03	25.5	Book3	0
5	Fantasy	ISBN04	25.5	Book2	2020
6	NULL	ISBN06	26.5	Book6	0

16 • `select * from bookstore.author_book;`

Result Grid		Filter Rows:		Edit:
author_id	book_id			
1	1			
2	1			
1	6			
3	6			

18 • `select * from bookstore.authors;`

Result Grid			Filter Rows:		E
id	birthday	name			
1	01-01-1992	author2			
2	02-02-1993	author1			
3	03-03-1994	author3			
4	04-04-1993	author4			

- **Delete Book by Id: Error Handling**

There will be no book deleted and database will not be updated when the book id does not exist.

https://localhost:8443/DeleteBookById/12

DELETE https://localhost:8443/DeleteBookById/12

Params Authorization Headers (10) Body Pre-request Script Tests Settings

Body Cookies (1) Headers (12) Test Results

Status: 404 Not Found Time: 35 ms Size: 531 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "timestamp": "2022-08-16T18:04:42.816+00:00",
3   "message": "No Book with Id : 12",
4   "details": "uri=/DeleteBookById/12"
5 }
```

1.7 Return Books by Author's Name (/ReturnBooksByAuthorName/{author_name})

- **Return Books by Author's Name: Basic Functionality**

User should be able to query all the books associated with an author by passing the Author's name as a parameter. We assume that the author's name is unique.

Initial Database State

14 • `select * from bookstore.books;`

Result Grid					
Filter Rows:					
id	genre	isbn	price	title	year
1	NULL	ISBN01	25.5	Book1	0
4	NULL	ISBN03	25.5	Book3	0
5	Fantasy	ISBN04	25.5	Book2	2020
6	NULL	ISBN06	26.5	Book6	0

16 • `select * from bookstore.author_book;`

Result Grid		Filter Rows:		Edit:
author_id	book_id			
1	1			
2	1			
1	6			
3	6			

18 • `select * from bookstore.authors;`

Result Grid			Filter Rows:		E
id	birthday	name			
1	01-01-1992	author2			
2	02-02-1993	author1			
3	03-03-1994	author3			
4	04-04-1993	author4			

Results

The pictures below show the result when the user queries for author2 (author_id = 1).

https://localhost:8443/ReturnBooksByAuthorName/author2

GET https://localhost:8443/ReturnBooksByAuthorName/author2

Params Authorization Headers (10) Body Pre-request Script Tests Settings

Body Cookies (1) Headers (12) Test Results

Status: 200 OK Time: 41 ms Size: 564 B Save Response

Pretty Raw Preview Visualize JSON

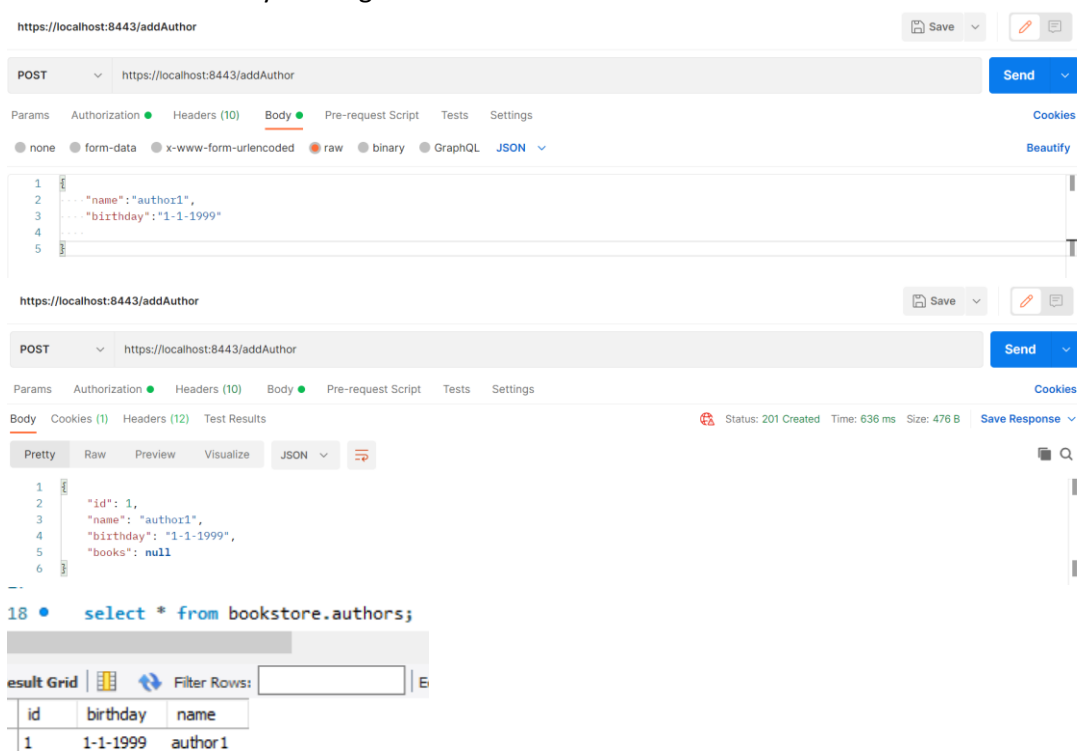
```
1 {
2   {
3     "id": 1,
4     "isbn": "ISBN01",
5     "title": "Book1",
6     "year": 0,
7     "price": 25.5,
8     "genre": null
9   },
10  {
11    "id": 6,
12    "isbn": "ISBN06",
13    "title": "Book6",
14    "year": 0,
15    "price": 26.5,
16    "genre": null
17  }
18 }
```

2.0 Author Controller

2.1 Add Author(/addAuthor)

- Add Author: Basic Functionality

User should be able to add new authors. However, the author to be added cannot have the same name as any existing authors.



```
POST https://localhost:8443/addAuthor
```

```
{
  "name": "author1",
  "birthday": "1-1-1999"
}
```

```
POST https://localhost:8443/addAuthor
```

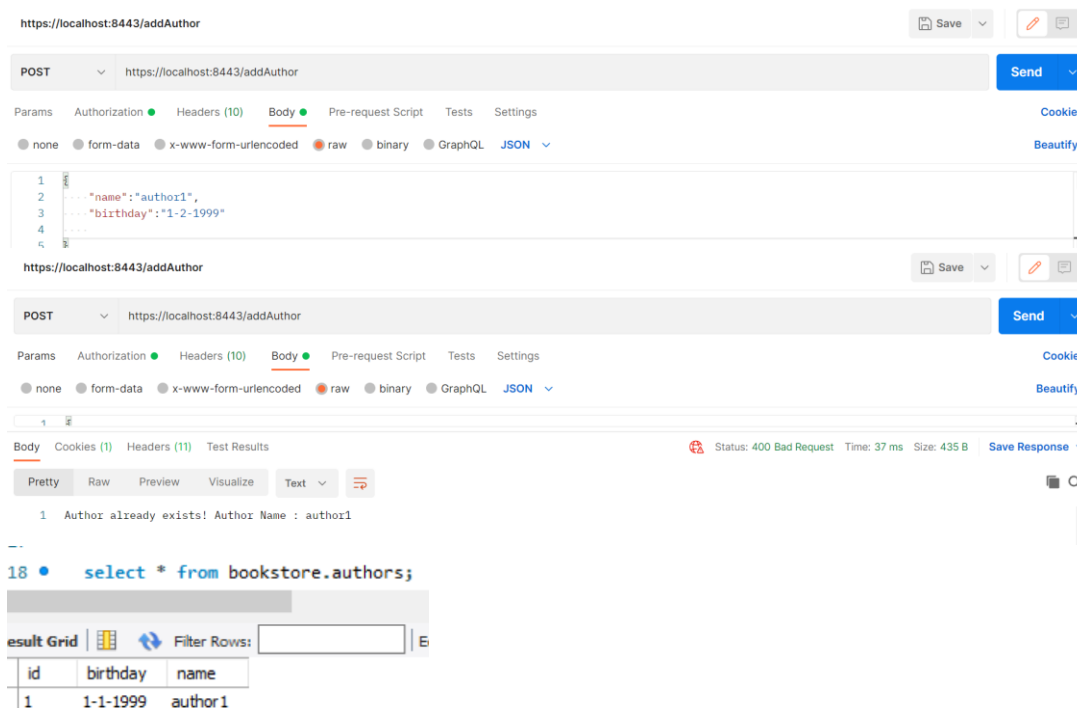
```
{
  "id": 1,
  "name": "author1",
  "birthday": "1-1-1999",
  "books": null
}
```

```
18 • select * from bookstore.authors;
```

id	birthday	name
1	1-1-1999	author1

■ Add Author: Scenarios

When user add an author with the same name as an existing author (even with different birthdates), the following error message will be thrown. The new author will not be added to database.



```
POST https://localhost:8443/addAuthor
```

```
{
  "name": "author1",
  "birthday": "1-2-1999"
}
```

```
POST https://localhost:8443/addAuthor
```

```
1 Author already exists! Author Name : author1
```

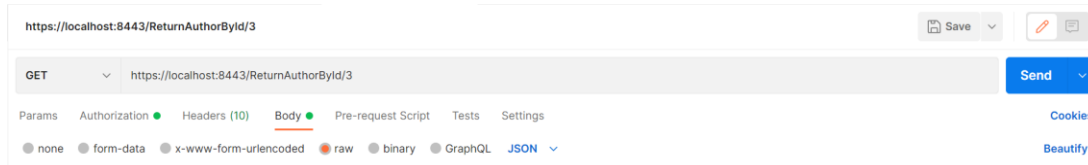
```
18 • select * from bookstore.authors;
```

id	birthday	name
1	1-1-1999	author1

2.2 Return Author by Id /ReturnAuthorById/{id})

- Return Author by Id: Basic Functionality

User should be able to query to get information on an author through providing an Author ID.



https://localhost:8443/ReturnAuthorById/3

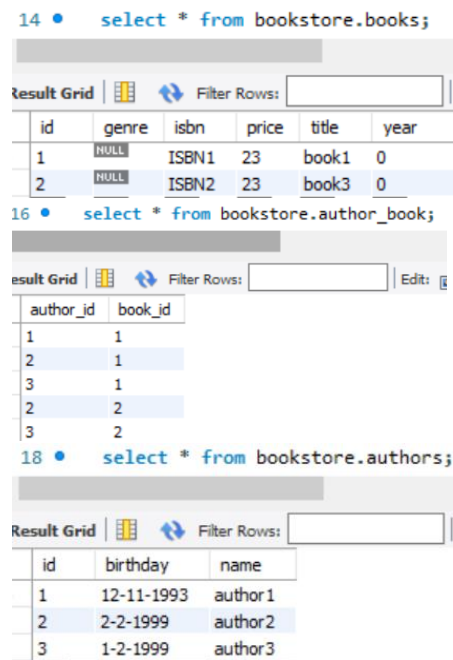
GET https://localhost:8443/ReturnAuthorById/3

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Save Send Cookies Beautify

Initial Database State



14 • select * from bookstore.books;

Result Grid Filter Rows:

id	genre	isbn	price	title	year
1	NULL	ISBN1	23	book1	0
2	NULL	ISBN2	23	book3	0

16 • select * from bookstore.author_book;

Result Grid Filter Rows: Edit:

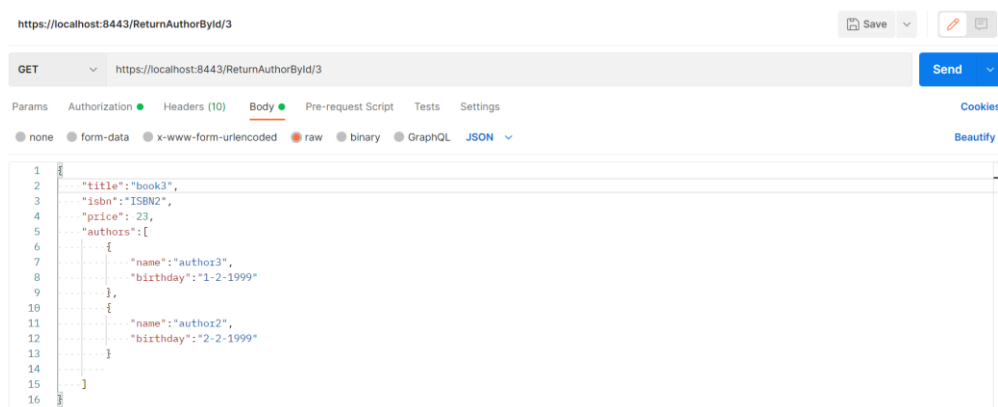
author_id	book_id
1	1
2	1
3	1
2	2
3	2

18 • select * from bookstore.authors;

Result Grid Filter Rows:

id	birthday	name
1	12-11-1993	author1
2	2-2-1999	author2
3	1-2-1999	author3

Results



https://localhost:8443/ReturnAuthorById/3

GET https://localhost:8443/ReturnAuthorById/3

Params Authorization Headers (10) Body Pre-request Script Tests Settings

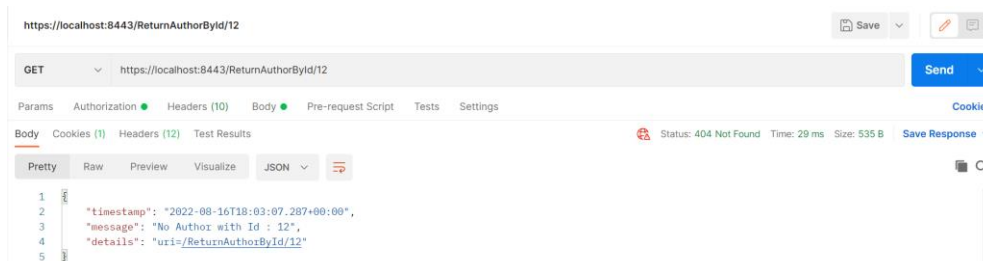
none form-data x-www-form-urlencoded raw binary GraphQL JSON

Save Send Cookies Beautify

```
1 {
2   "title": "book3",
3   "isbn": "ISBN2",
4   "price": 23,
5   "authors": [
6     {
7       "name": "author3",
8       "birthday": "1-2-1999"
9     },
10    {
11      "name": "author2",
12      "birthday": "2-2-1999"
13    }
14  ]
15 }
16
```

- Return Author by Id: Error Handling

There will be no content returned when the author Id does not exist.



https://localhost:8443/ReturnAuthorById/12

GET https://localhost:8443/ReturnAuthorById/12

Params Authorization Headers (10) Body Pre-request Script Tests Settings

Body Cookies (1) Headers (12) Test Results

Status: 404 Not Found Time: 29 ms Size: 535 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "timestamp": "2022-08-16T18:03:07.287+00:00",
3   "message": "No Author with Id : 12",
4   "details": "/ReturnAuthorById/12"
5 }
```

```
18 • select * from bookstore.authors;
```

Result Grid

Filter Rows:

id	birthday	name
1	12-11-1993	author1
2	2-2-1999	author2
3	1-2-1999	author3

Notes

Due to time constraint, only manual testing has been implemented. Junit Test could be later implemented.