

Source code: <https://github.com/sidzhik/Bookshop>

Needed dependencies while I was creating project via spring boot:

- do wyświetlenia na stronie internetowej <http://localhost:8080/>...

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

- do zarządzania i pracy z bazą danych

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
</dependency>
```

General class for running project, which was created by spring constructor:

```
@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

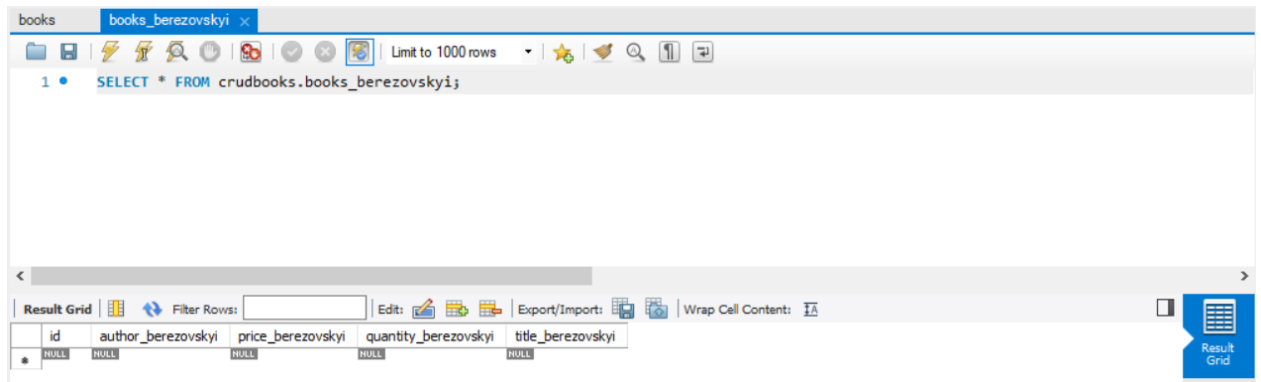
}
```

Created classes and interface for managing with our data (CRUD create read update delete), creating table with variables:

- BooksController (manage)
- Books (create table to database and set/get variable)
- BooksRepo (Interface, which extends JpaRepository, which consists of JPA methods for work with DataBase)

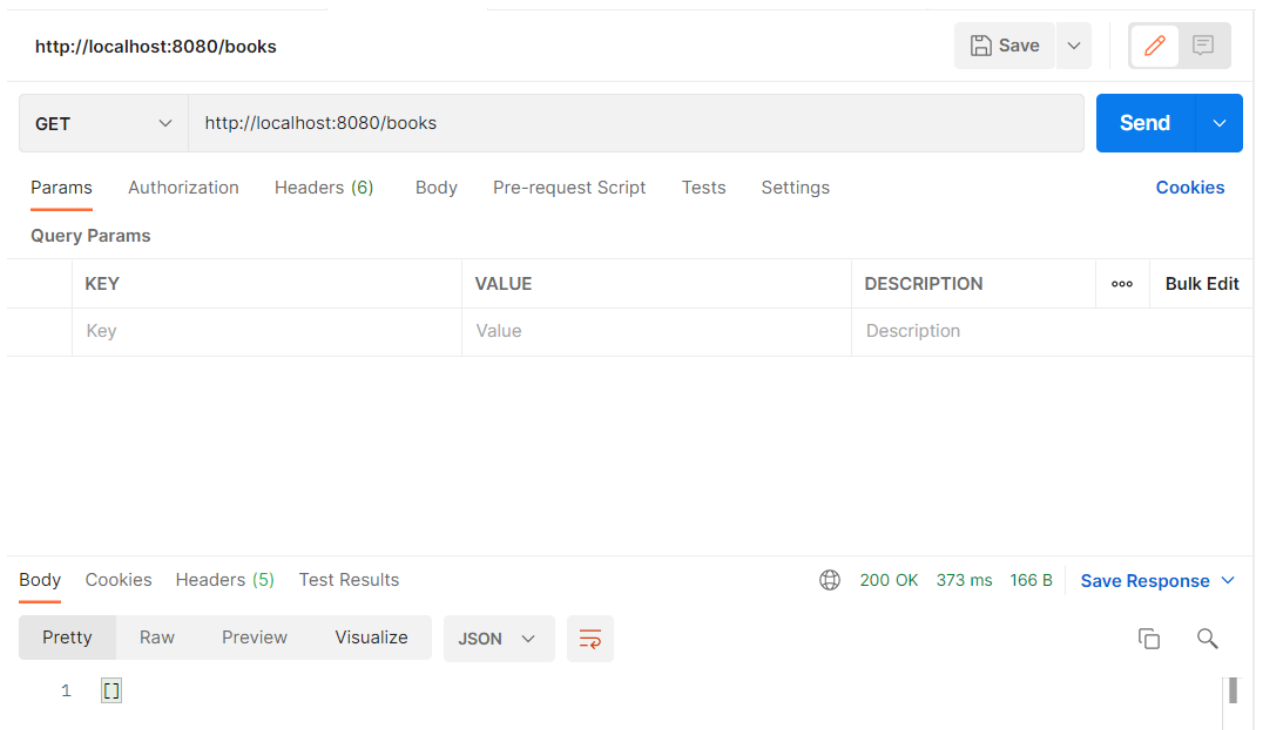
Podłączenie do bazy danych:

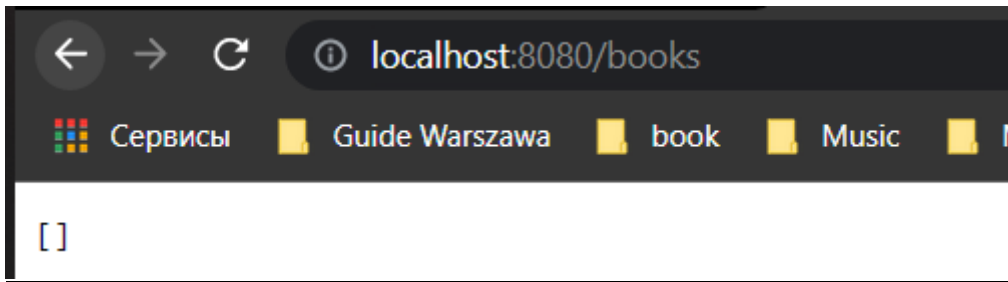
Korzystałem z WorkBench. Stworzyłem nową bazę danych, za pomocy stworzonej wcześniej klasy Books z adnotacją @Entity i wszystkimi zadeklarowanymi zmiennymi, po urochomieniu była stworzona tablica



Testowanie funkcjonalne przez Postman (Get,Post,Put,Delete)

Get:





Rzeczywiście, że jakichkolwiek danych jeszcze nie ma, ale jest pusta tabela []

Post:

A screenshot of a REST client interface. The top bar shows the URL 'http://localhost:8080/save' and a 'Save' button. Below this, the method is set to 'POST' and the URL is 'http://localhost:8080/save'. The 'Body' tab is selected, showing a JSON payload:

```
{ 1: { 2: "title_berezovskyi": "Rich dad poor dad", 3: "author_berezovskyi": "Rober Kiyosaki", 4: "quantity_berezovskyi": 50, 5: "price_berezovskyi": 48 6: }
```

 The bottom section shows the response status '200 OK' and a message 'Dane zostało zapisane'.A screenshot of a database client interface. The top bar shows a SQL query: 'SELECT * FROM crudbooks.books_berezovskyi;'. Below the query, the results are displayed in a table with columns: 'id', 'author_berezovskyi', 'price_berezovskyi', 'quantity_berezovskyi', and 'title_berezovskyi'. The table contains one row with the following data:

id	author_berezovskyi	price_berezovskyi	quantity_berezovskyi	title_berezovskyi
7	Rober Kiyosaki	48	50	Rich dad poor dad

[{"id":7,"titleBerezovskyi":"Rich dad poor dad","authorBerezovskyi":"Rober Kiyosaki","quantityBerezovskyi":50,"priceBerezovskyi":48}]

Put:

http://localhost:8080/edit/8

Save

PUT

http://localhost:8080/edit/8

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

1

2

3

4

5

6

....."titleBerezovskyi": "Rich-dad-poor-dad",
....."authorBerezovskyi": "Robert-Kiyosaki",
....."quantityBerezovskyi": 100,
....."priceBerezovskyi": 60

Body

Cookies

Headers (5)

Test Results

200 OK

46 ms

187 B

Save Response

Pretty

Raw

Preview

Visualize

Text

1

Dane zostało edytowane

, "quantityBerezovskyi": 100, "priceBerezovskyi": 60}]

price_berezovskyi	quantity_berezovskyi
60	100

Delete:

Save  

Send 

Cookies

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

200 OK 62 ms 196 B Save Response

```
1 • SELECT * FROM crudbooks.books_berezovskyi;
```