Maids.cc Project

Book:

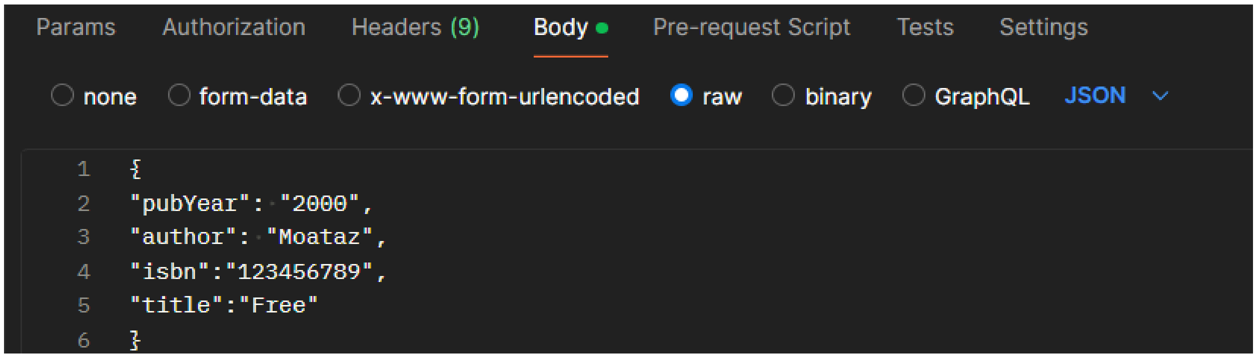
The class contains (bookId, publish year, author, ISBN, title)

Endpoints:

- GET: http://localhost:8082/books:
  This url is used to return the list of books stored in the database.

- GET: http://localhost:8082/books/{id}
  This url is used to retrieve a book by passing its id.

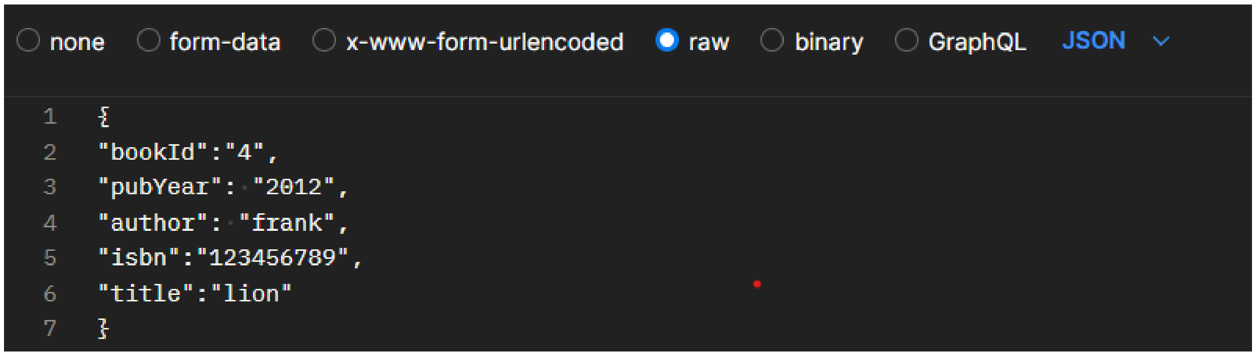
- POST: http://localhost:8082/books
  This url is used to add a new book by inserting its attributes in the request body (JSON).



Body: raw, JSON

```
{
"pubYear": "2000",
"author": "Moataz",
"isbn":"123456789",
"title":"Free"
}
```

- PUT: http://localhost:8082/books/{id}
  This url is used to update an existing book by passing its id and then edit its attributes in the request body.



Body: raw, JSON

```
{
"bookId":"4",
"pubYear": "2012",
"author": "frank",
"isbn":"123456789",
"title":"lion"
}
```

- DELETE: http://localhost:8082/books/{id}
  This url is used to delete an existing book by passing its id.

Patron:

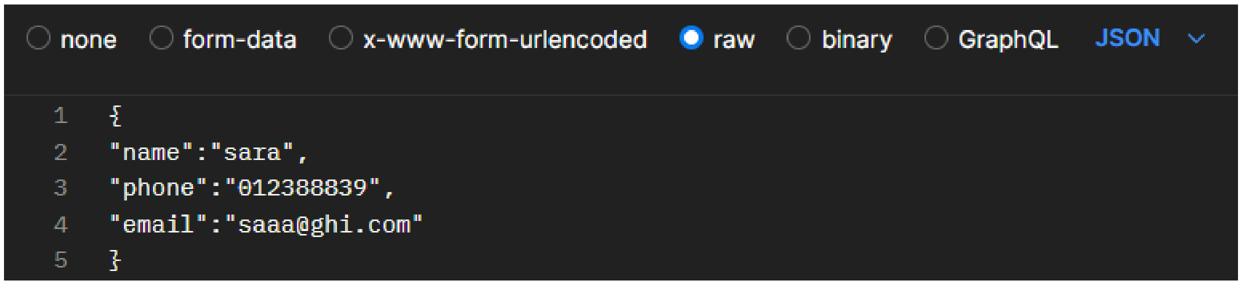
This class contains (name, email, phone, patronId)

Endpoints:

- GET: http://localhost:8082/patrons
  This url is used to return all the patrons stored in the database.

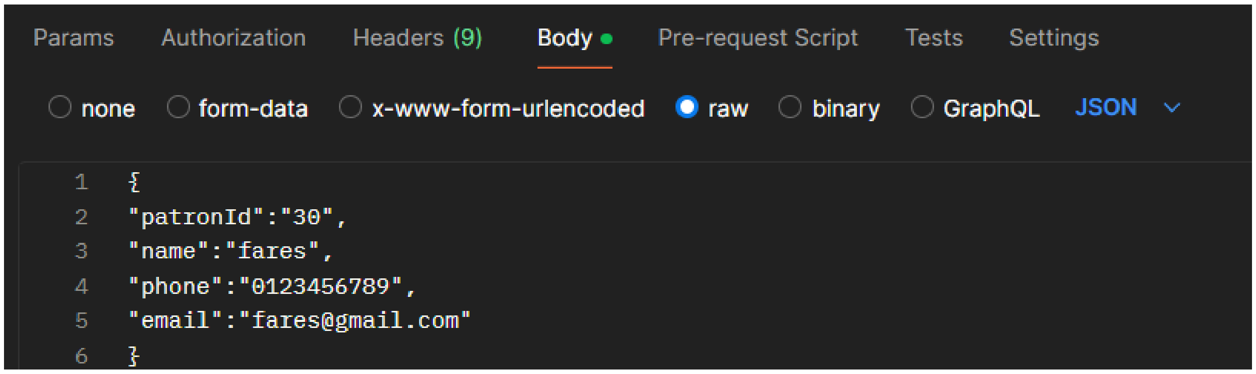
- GET: http://localhost:8082/patrons/{id}
  This url is used to retrieve an existing patron by passing its id.

- POST: http://localhost:8082/patrons
  This url is used to add a new patron by inserting its attributes in the request body (JSON).



```
{
"name":"sara",
"phone":"012388839",
"email":"saaa@ghi.com"
}
```

- PUT: http://localhost:8082/patrons/{id}
  This url is used to update an existing patron by passing its id and then edit its attributes in the request body.



```
{
"patronId":"30",
"name":"fares",
"phone":"0123456789",
"email":"fares@gmail.com"
}
```

- DELETE: http://localhost:8082/patrons/{id}
  This url is used to delete an existing patron by passing its id.

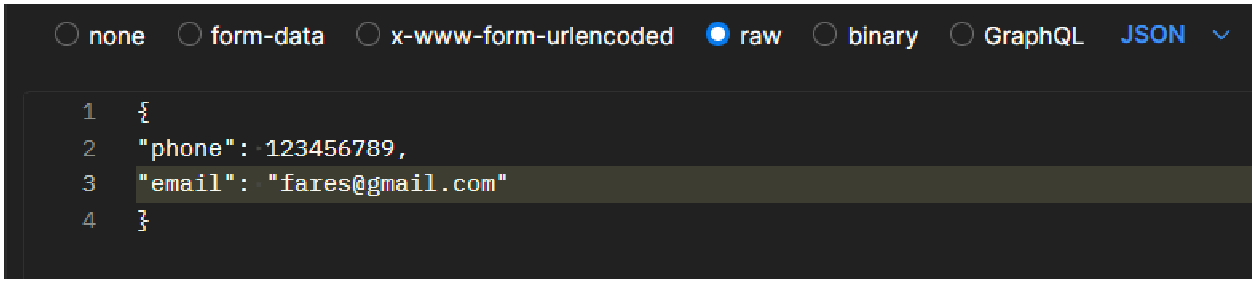
Borrowing Record:

This class contains (instance of book, instance of patron, id, borrow date, return date, due date, returned)

Returned: is a Boolean variable to determine whether the borrowed book returned or not

Endpoints:

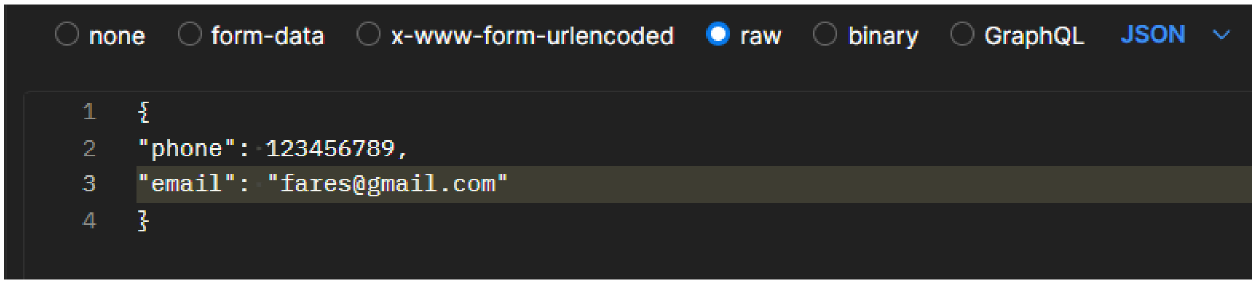
- POST: http://localhost:8082/borrow/{bookID}/patron/{patronID}
  This url is used to record the process of a patron borrowing a book by passing book id and patron id respectively.
  The request requires to insert the email and phone of the patron (Log in info) in the request body to authenticate the patron's information and borrowing process.



none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
"phone": 123456789,
"email": "fares@gmail.com"
}
```

- PUT: http://localhost:8082/return/{bookID}/patron/{patronID}
  This url is used to record the return date of the borrowed book by passing book id and patron id respectively.
  The request requires to insert the email and phone of the patron (Log in info) in the request body to authenticate the patron's information and borrowing process.



none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
"phone": 123456789,
"email": "fares@gmail.com"
}
```

Exception Handling:

The project implements exception handling by validating book and patron existence and their variables. Creation of custom exceptions to handle every case.

Testing:

The project uses Mokito library to create a unit test for each method in each service.

Caching:

Caching is implemented by adding (@cachable) annotation above some methods for patron and book services (Get by Id, Add, Edit, Delete), to ensure the caching of the returned values of these methods to improve the system's performance.

Transaction Management:

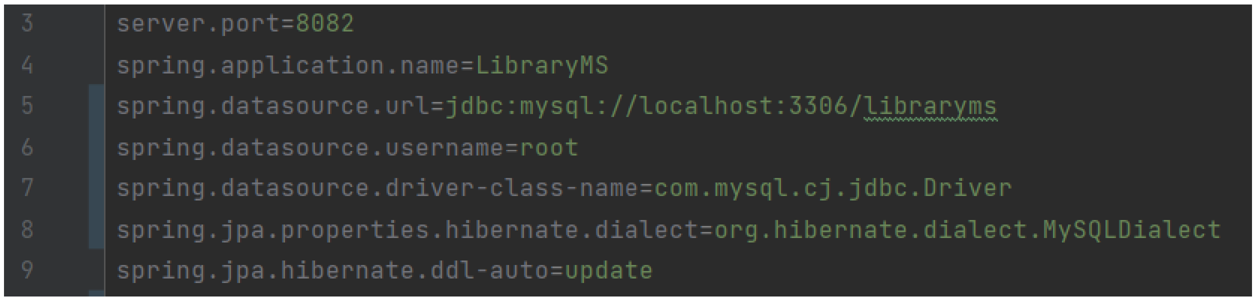
Transaction Management is implemented by adding (@transactional) annotation before some methods for patron, book and borrowing record services (Add, Edit, Borrow, Return) to ensure that if the method fails for any reason the whole transaction will rollback and not be added in the database.

Application Properties:

The project is hosted on "localhost port : 8082" which can be changed in the application properties from "server.port".

Database connection properties:

- The project is using MYSQL database
- Database name: LibraryMS
- To connect the project to your database create a database named "LibraryMS" and change the log in credentials (username, password) that are found in the application properties



```
server.port=8082
spring.application.name=LibraryMS
spring.datasource.url=jdbc:mysql://localhost:3306/libraryms
spring.datasource.username=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=update
```