try

tries to execute problemetic code

catch

exception handler

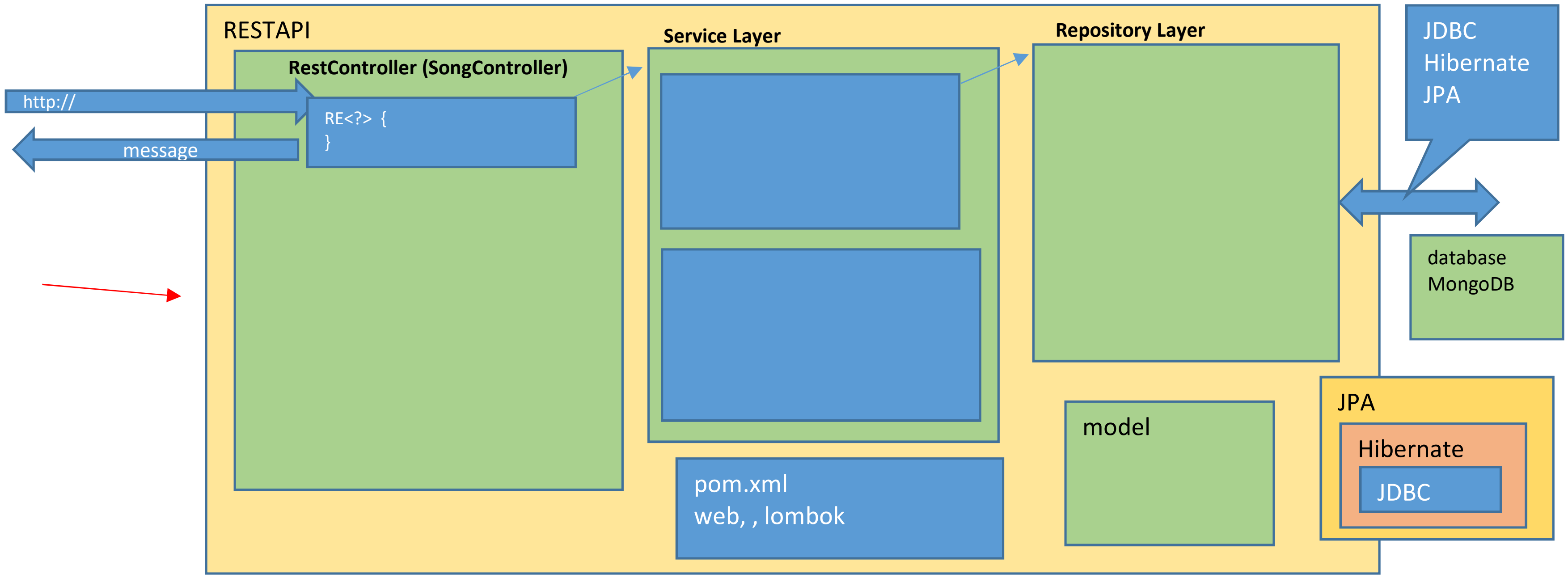
finally

throw

to throw object of any Exception class

throws

to declare/define a method as exception throwable meth



controller <-> service <-> repo <-> Db

response
data+status
exp message+status

mongoDb

spring-data-jpa
Repository
CrudRepository h2
JpaRepository mysql

	Song
String	songid
String	name
String	artist
String	duration
int	rating

	Book
String	bookId
String	bookName
String	subjet
Author	author
int	price
int	stock

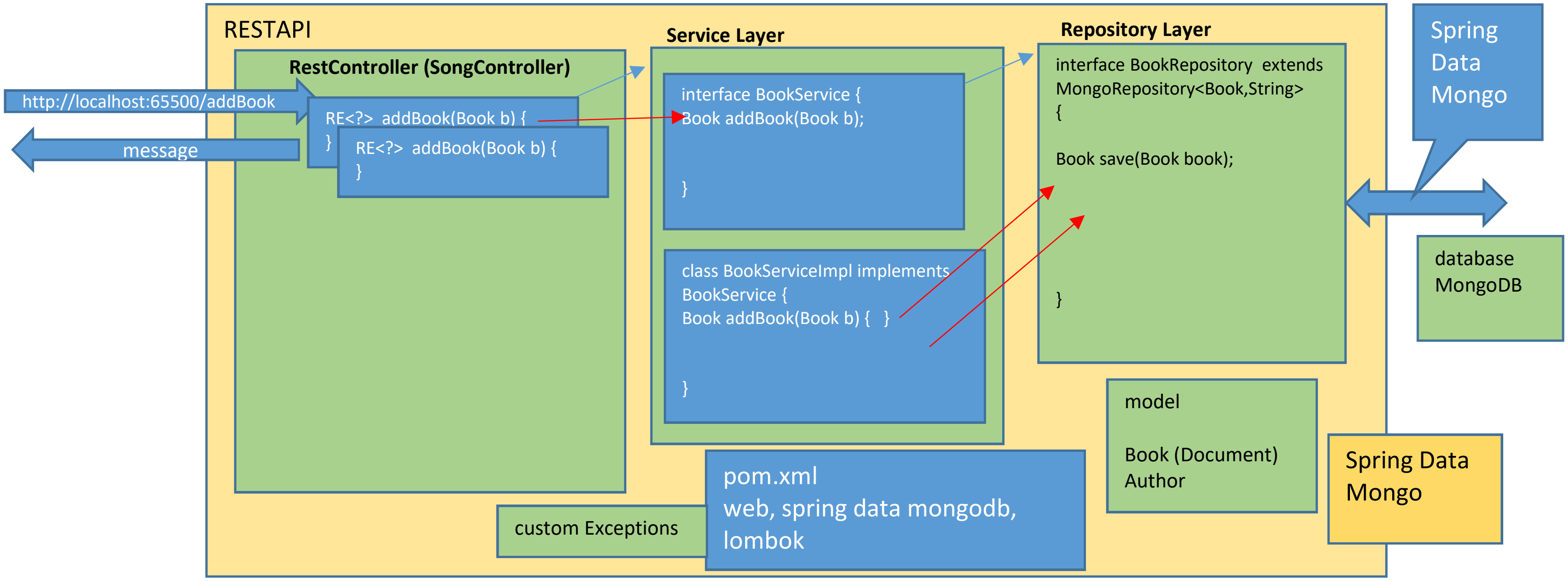
Author

authorName
address
age

Book HAS Author

@Entity	@Document
@Id	@Id

```
{
bookId:"BK0001",
name:"CSS in HTML",
subject:"HTML",
author: { authorName:"Raju",address:"Pune",age:31},
price:345,
stock:45
}
```



RESTAPI

RestController (SongController)

http://localhost:65500/addBook

message

```
RE<?>  addBook(Book b) {
}
```

```
RE<?>  addBook(Book b) {
}
```

Service Layer

```
interface BookService {
Book addBook(Book b);


}
```

```
class BookServiceImpl implements
BookService {
Book addBook(Book b) {   }


}
```

Repository Layer

```
interface BookRepository  extends
MongoRepository<Book,String>
{

Book save(Book book);




}
```

Spring Data Mongo

database MongoDB

model

Book (Document)
Author

Spring Data Mongo

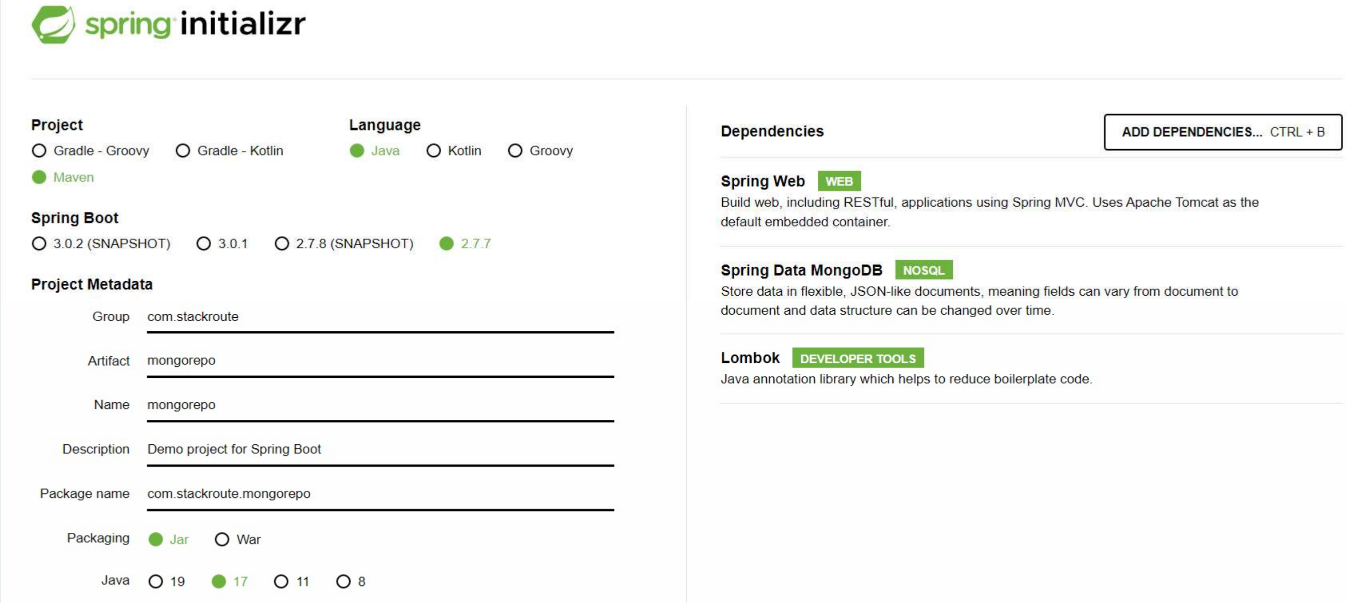
custom Exceptions

pom.xml
web, spring data mongodb,
lombok

Steps to create RESTFULL API appication using MongoDB as Data server

Task 1 : inserting new book record

Step 1 Create new springboot application in spring initializer adding required dependencies



Project

○ Gradle - Groovy ○ Gradle - Kotlin ● Maven

Language

● Java ○ Kotlin ○ Groovy

Spring Boot

○ 3.0.2 (SNAPSHOT) ○ 3.0.1 ○ 2.7.8 (SNAPSHOT) ● 2.7.7

Project Metadata

Group: com.stackroute

Artifact: mongorepo

Name: mongorepo

Description: Demo project for Spring Boot

Package name: com.stackroute.mongorepo

Packaging: ● Jar ○ War

Java: ○ 19 ● 17 ○ 11 ○ 8

Dependencies

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

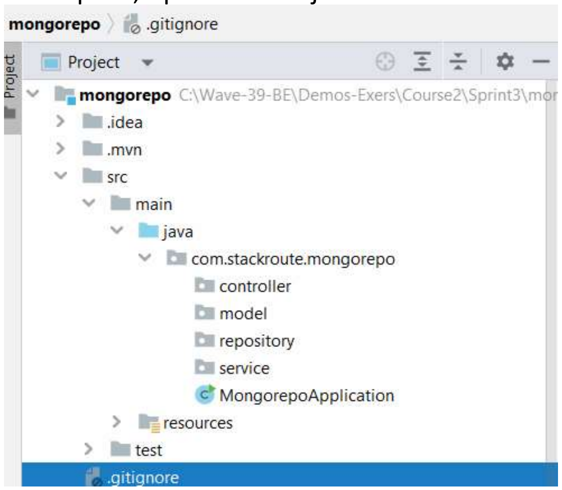
Spring Data MongoDB NOSQL
Store data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time.

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

download, extract, copy to workspace, open in intellij

check settings if required

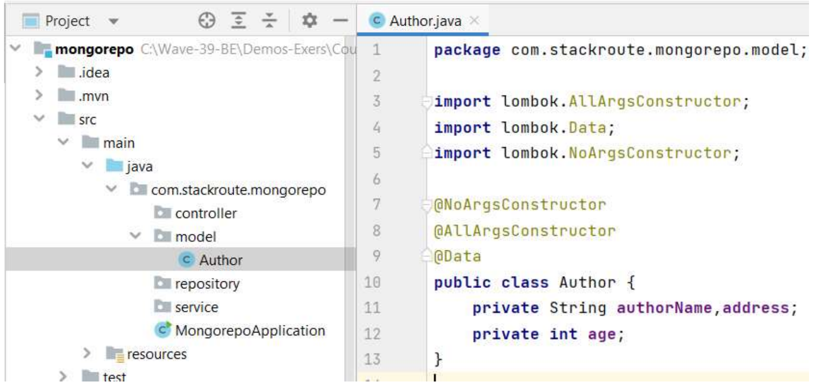
add required packages



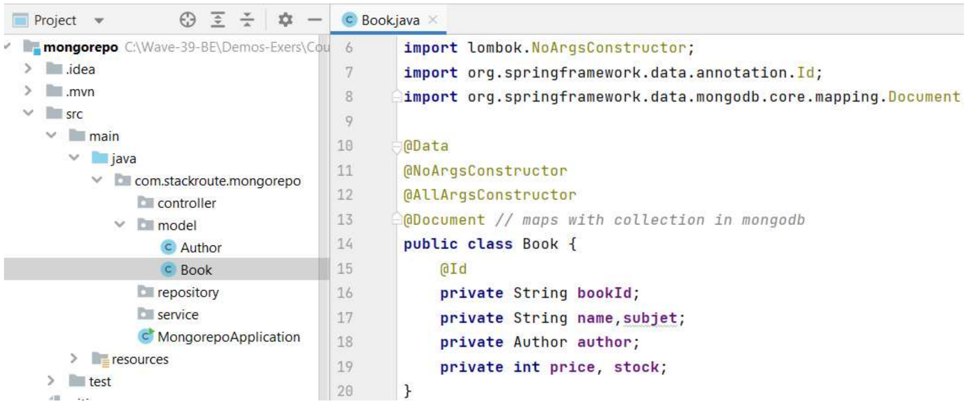
```
mongorepo C:\Wave-39-BE\Demos-Exers\Course2\Sprint3\mor
  .idea
  .mvn
  src
    main
      java
        com.stackroute.mongorepo
          controller
          model
          repository
          service
          MongorepoApplication
      resources
    test
  .gitignore
```

Step 2 Create model classes

Book HAS Author

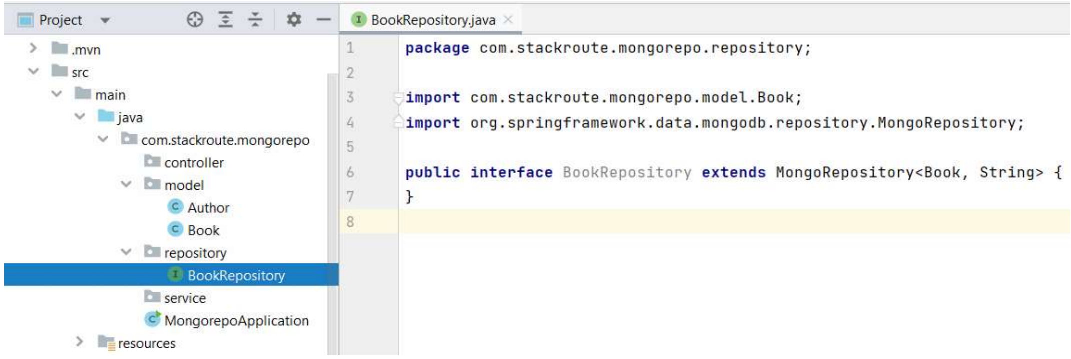


```java
package com.stackroute.mongorepo.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@NoArgsConstructor
@AllArgsConstructor
@Data
public class Author {
    private String authorName,address;
    private int age;
}
```



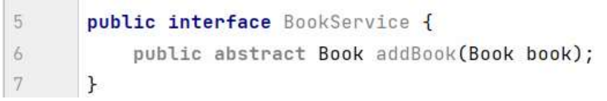
```java
import lombok.NoArgsConstructor;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document

@Data
@NoArgsConstructor
@AllArgsConstructor
@Document // maps with collection in mongodb
public class Book {
    @Id
    private String bookId;
    private String name,subjet;
    private Author author;
    private int price, stock;
}
```

Step 3 Define repository

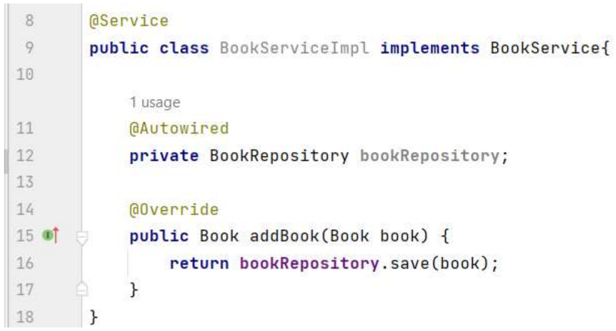


```java
package com.stackroute.mongorepo.repository;

import com.stackroute.mongorepo.model.Book;
import org.springframework.data.mongodb.repository.MongoRepository;

public interface BookRepository extends MongoRepository<Book, String> {
}
```

Step 4 Define service layer



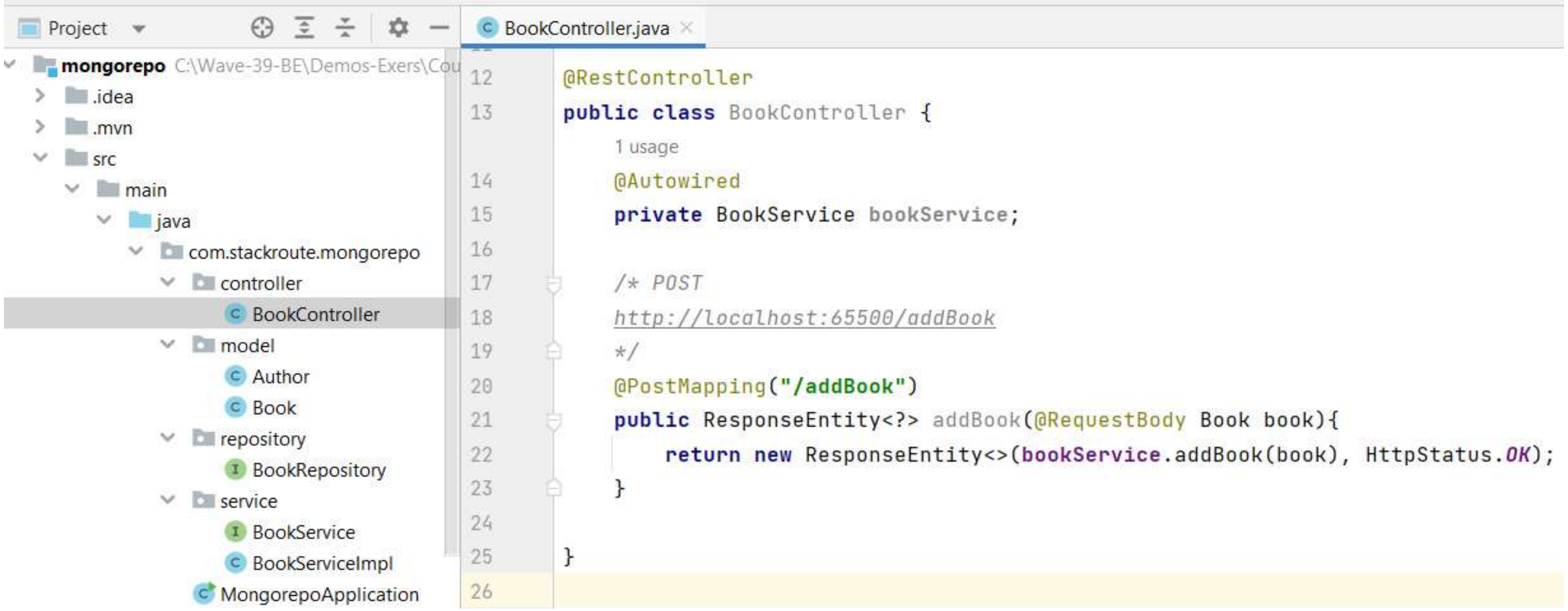
```java
public interface BookService {
    public abstract Book addBook(Book book);
}
```



```java
@Service
public class BookServiceImpl implements BookService{

    1 usage
    @Autowired
    private BookRepository bookRepository;

    @Override
    public Book addBook(Book book) {
        return bookRepository.save(book);
    }
}
```

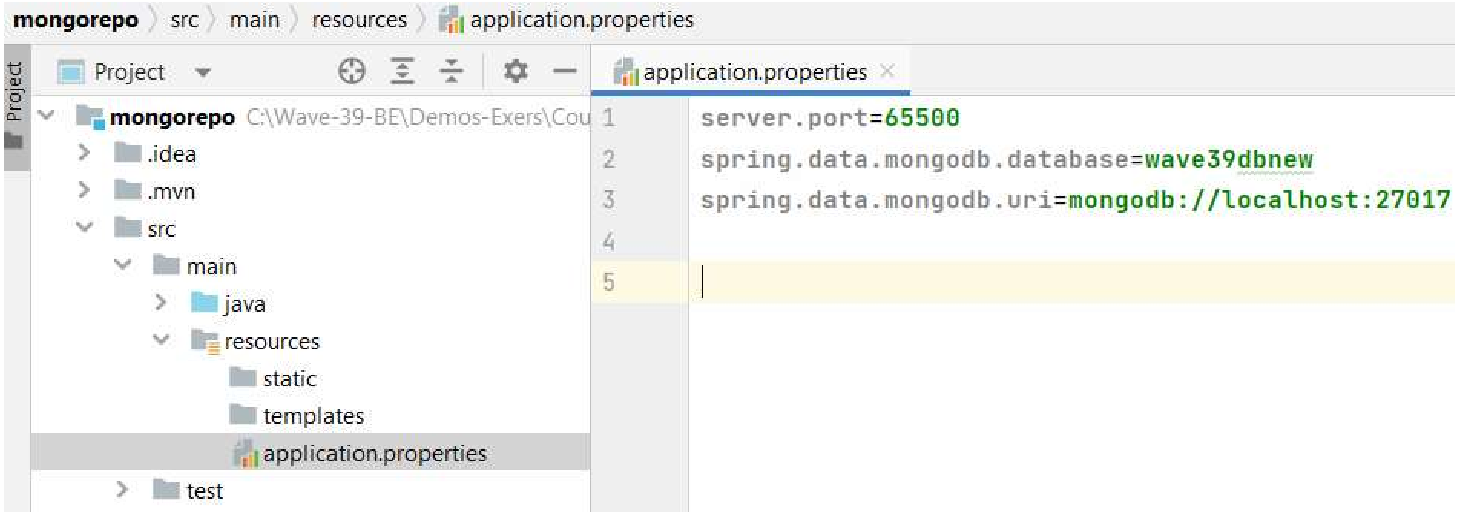
Step 5 Define controller layer

with request handler method



```java
@RestController
public class BookController {
    1 usage
    @Autowired
    private BookService bookService;

    /* POST
    http://localhost:65500/addBook
    */
    @PostMapping("/addBook")
    public ResponseEntity<?> addBook(@RequestBody Book book){
        return new ResponseEntity<>(bookService.addBook(book), HttpStatus.OK);
    }

}
```

Step 6 edit application.properties



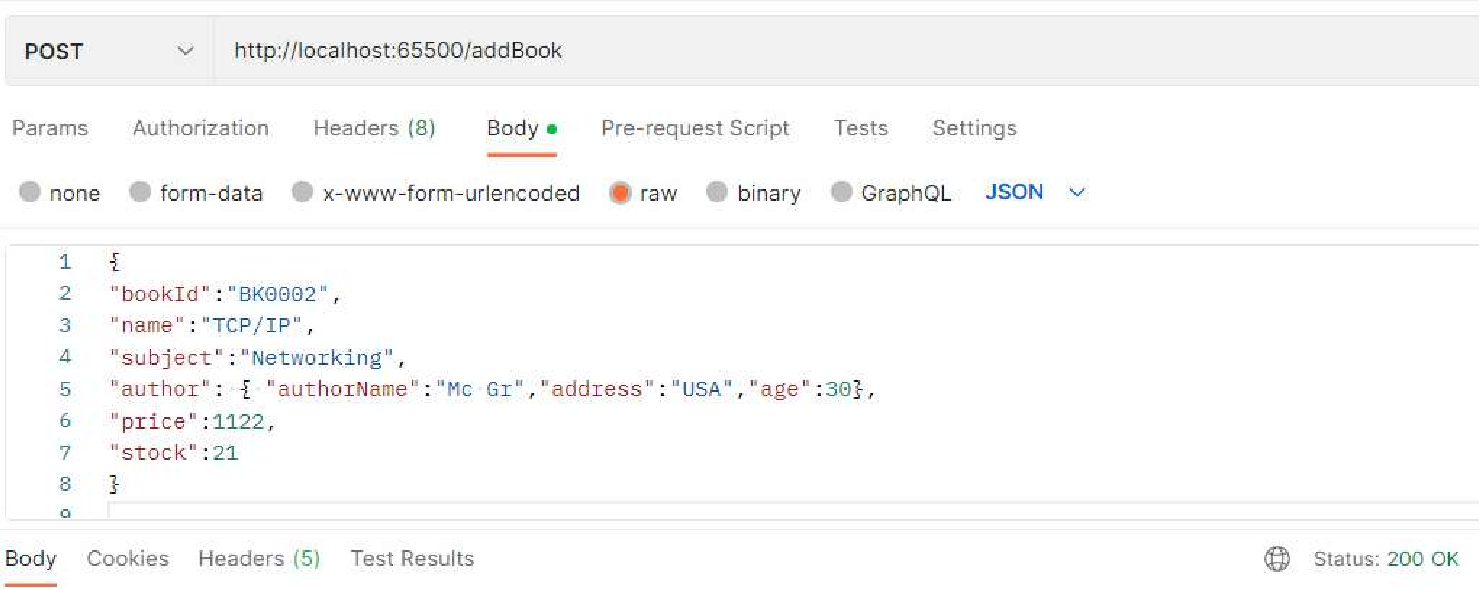
```
server.port=65500
spring.data.mongodb.database=wave39dbnew
spring.data.mongodb.uri=mongodb://localhost:27017

```

run application

open postmat

post one book record

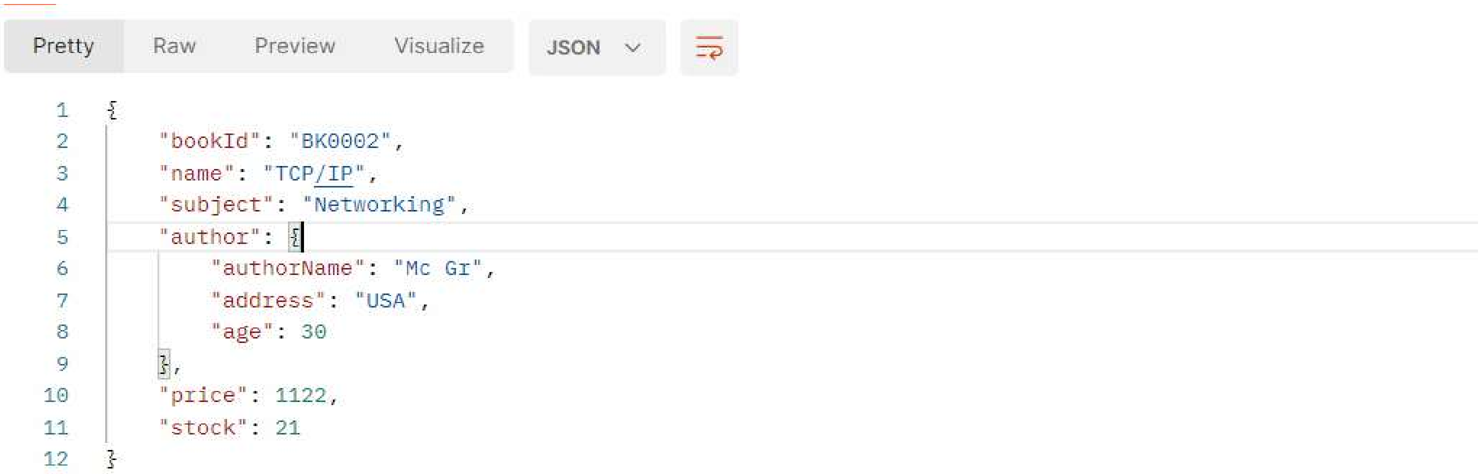
check in mongodb



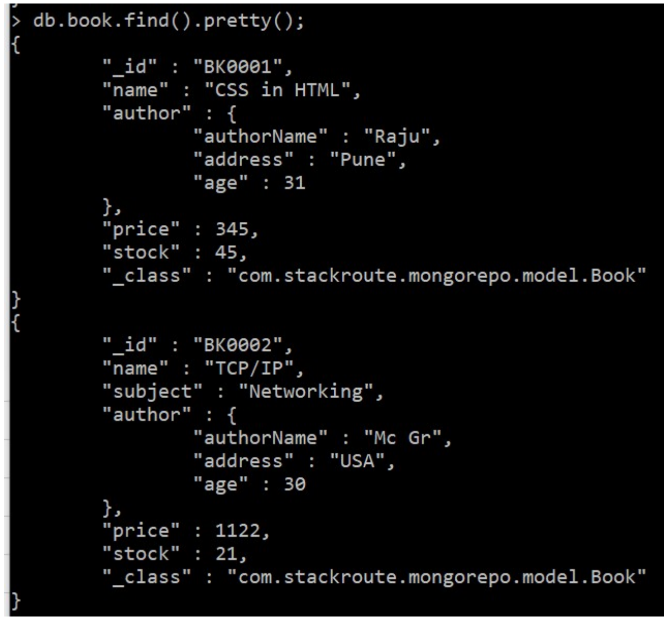
POST http://localhost:65500/addBook

```
{
"bookId":"BK0002",
"name":"TCP/IP",
"subject":"Networking",
"author": { "authorName":"Mc Gr","address":"USA","age":30},
"price":1122,
"stock":21
}
```

Status: 200 OK

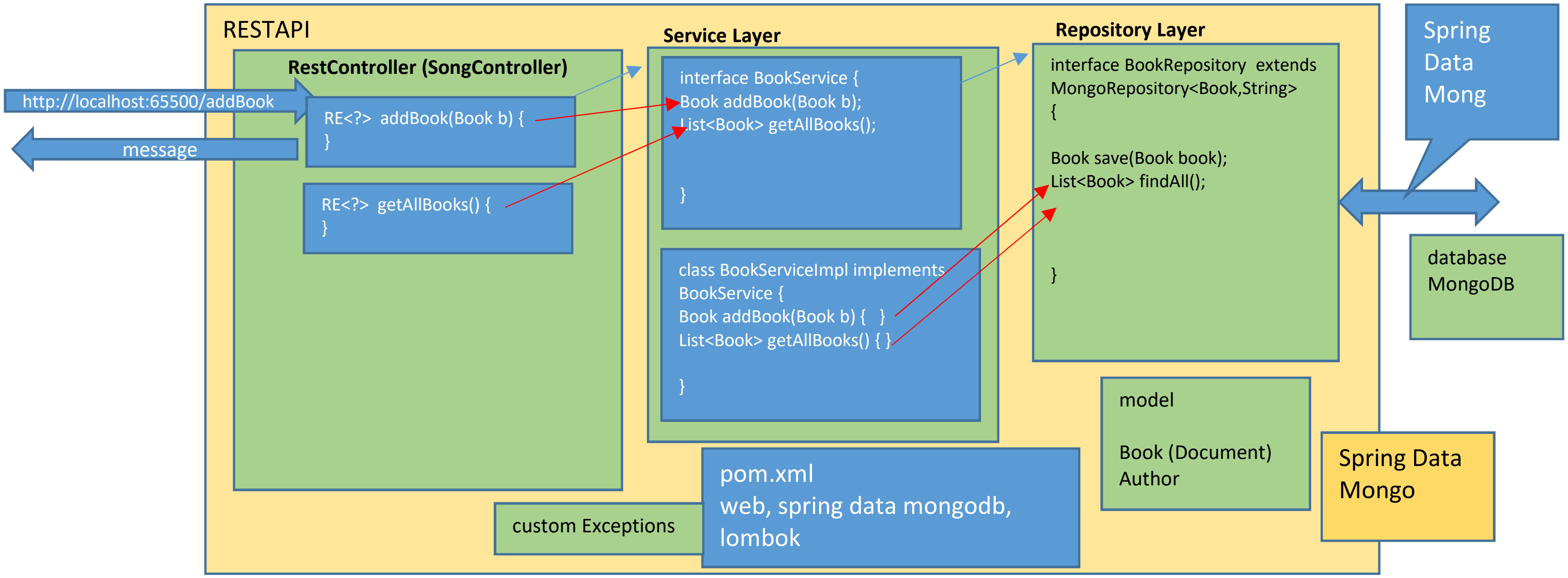


```
{
    "bookId": "BK0002",
    "name": "TCP/IP",
    "subject": "Networking",
    "author": {
        "authorName": "Mc Gr",
        "address": "USA",
        "age": 30
    },
    "price": 1122,
    "stock": 21
}
```



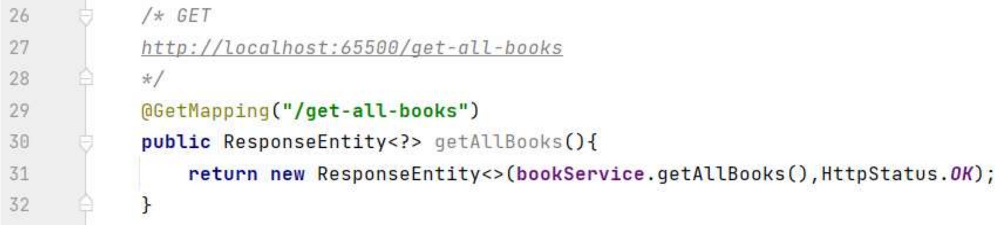
```
> db.book.find().pretty();
{
        "_id" : "BK0001",
        "name" : "CSS in HTML",
        "author" : {
                "authorName" : "Raju",
                "address" : "Pune",
                "age" : 31
        },
        "price" : 345,
        "stock" : 45,
        "_class" : "com.stackroute.mongorepo.model.Book"
}
{
        "_id" : "BK0002",
        "name" : "TCP/IP",
        "subject" : "Networking",
        "author" : {
                "authorName" : "Mc Gr",
                "address" : "USA",
                "age" : 30
        },
        "price" : 1122,
        "stock" : 21,
        "_class" : "com.stackroute.mongorepo.model.Book"
}
```

Task 2 : get all records

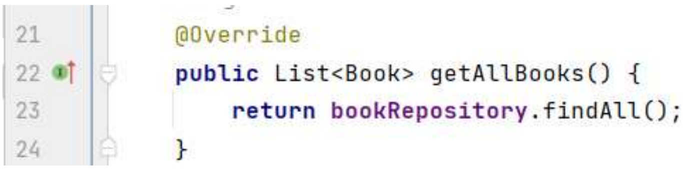


Step 1 service

Step 2 controller

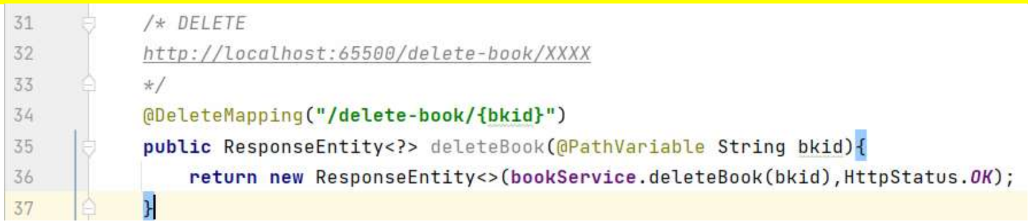


```
/* GET
http://localhost:65500/get-all-books
*/
@GetMapping("/get-all-books")
public ResponseEntity<?> getAllBooks(){
    return new ResponseEntity<>(bookService.getAllBooks(),HttpStatus.OK);
}
```

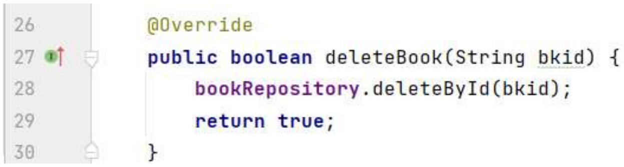


```
@Override
public List<Book> getAllBooks() {
    return bookRepository.findAll();
}
```

Task 3 : Delete book by id

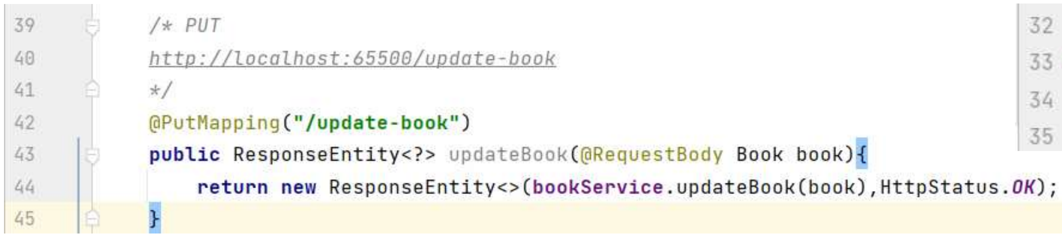


```
/* DELETE
http://localhost:65500/delete-book/XXXX
*/
@DeleteMapping("/delete-book/{bkid}")
public ResponseEntity<?> deleteBook(@PathVariable String bkid){
    return new ResponseEntity<>(bookService.deleteBook(bkid),HttpStatus.OK);
}
```

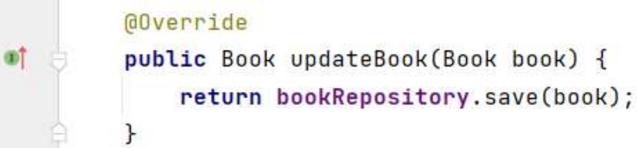


```
@Override
public boolean deleteBook(String bkid) {
    bookRepository.deleteById(bkid);
    return true;
}
```

Task 4 : Update book



```java
/* PUT
http://localhost:65500/update-book
*/
@PutMapping("/update-book")
public ResponseEntity<?> updateBook(@RequestBody Book book){
    return new ResponseEntity<>(bookService.updateBook(book),HttpStatus.OK);
}
```

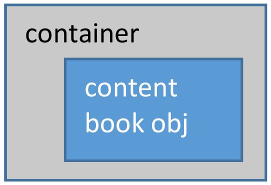


```java
@Override
public Book updateBook(Book book) {
    return bookRepository.save(book);
}
```

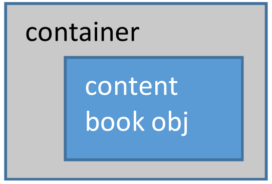
Task 5 : get book by Id

in repsository

Optional<Book> findById(String id);



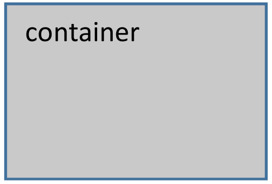
if content existing



container.isPresent
TRUE

container.isEmpty
FALSE

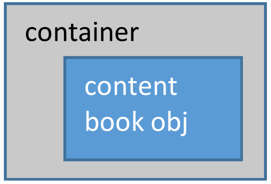
if content not existing



container.isPresent
FALSE

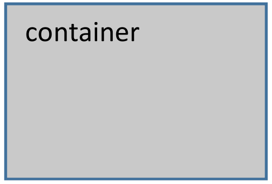
container.isEmpty
TRUE

if content existing

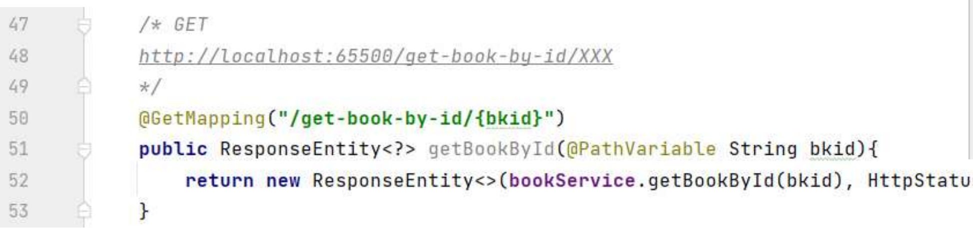


container.get()
returns content object

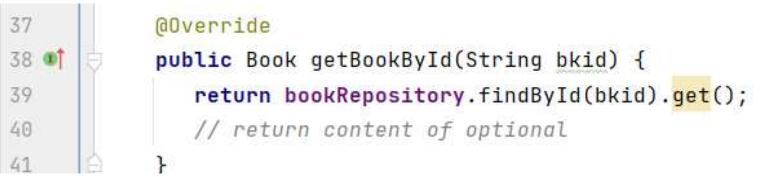
if content not existing



container.get()
throws exception

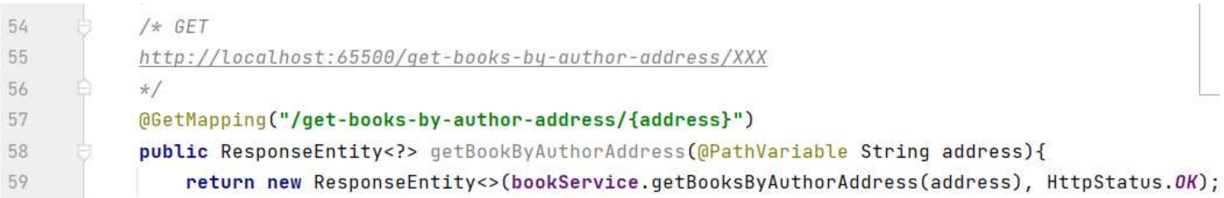


```java
/* GET
http://localhost:65500/get-book-by-id/XXX
*/
@GetMapping("/get-book-by-id/{bkid}")
public ResponseEntity<?> getBookById(@PathVariable String bkid){
    return new ResponseEntity<>(bookService.getBookById(bkid), HttpStatus.OK);
}
```

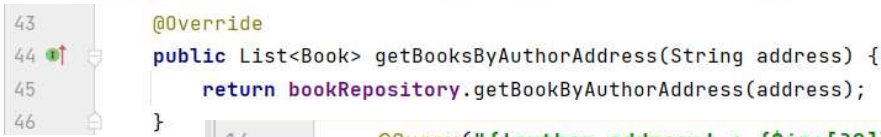


```java
@Override
public Book getBookById(String bkid) {
    return bookRepository.findById(bkid).get();
    // return content of optional
}
```

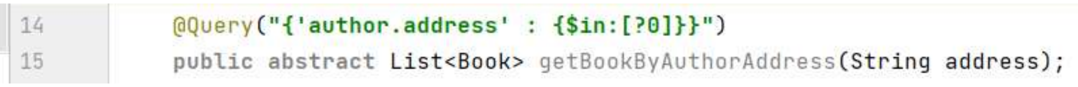
Custom method to get book details based on author.city



```java
/* GET
http://localhost:65500/get-books-by-author-address/XXX
*/
@GetMapping("/get-books-by-author-address/{address}")
public ResponseEntity<?> getBookByAuthorAddress(@PathVariable String address){
    return new ResponseEntity<>(bookService.getBooksByAuthorAddress(address), HttpStatus.OK);
}
```



```java
@Override
public List<Book> getBooksByAuthorAddress(String address) {
    return bookRepository.getBookByAuthorAddress(address);
}
```



```java
@Query("{'author.address' : {$in:[?0]}}")
public abstract List<Book> getBookByAuthorAddress(String address);
```

Add custom exception

in service

getAllBooks()   no need to implement exception

addBook(book object)   BookAlreadyExistingException

has to insert book record if object.id not existis in db   message, httpstatus

else, throw exception

deleteBook(bookid)   BookNotFoundException

has to delete book if bookid based record found in db   message, httpstatus

else, throw exception

updateBook(book object)   BookNotFoundException

has to update book if object.id based record found in db   message, httpstatus

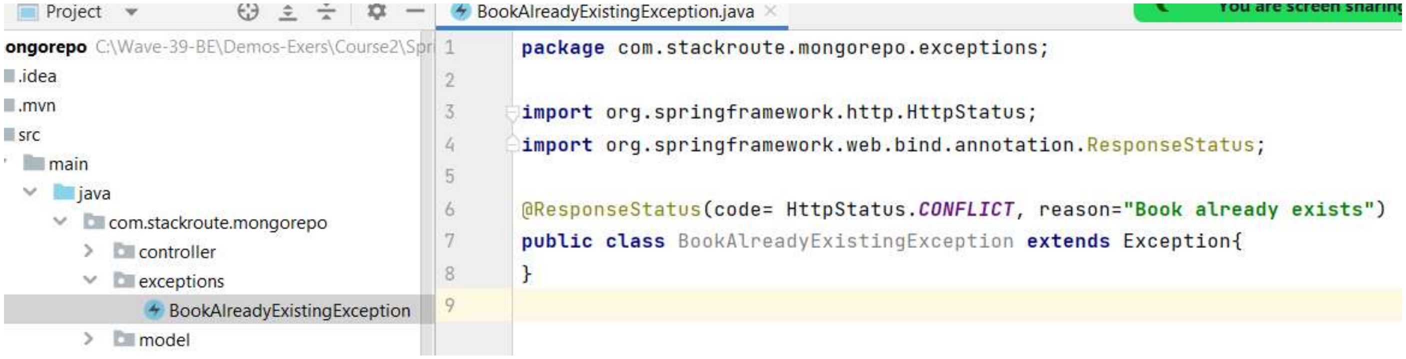
else, throw exception

getBookById(bookid)   BookNotFoundException

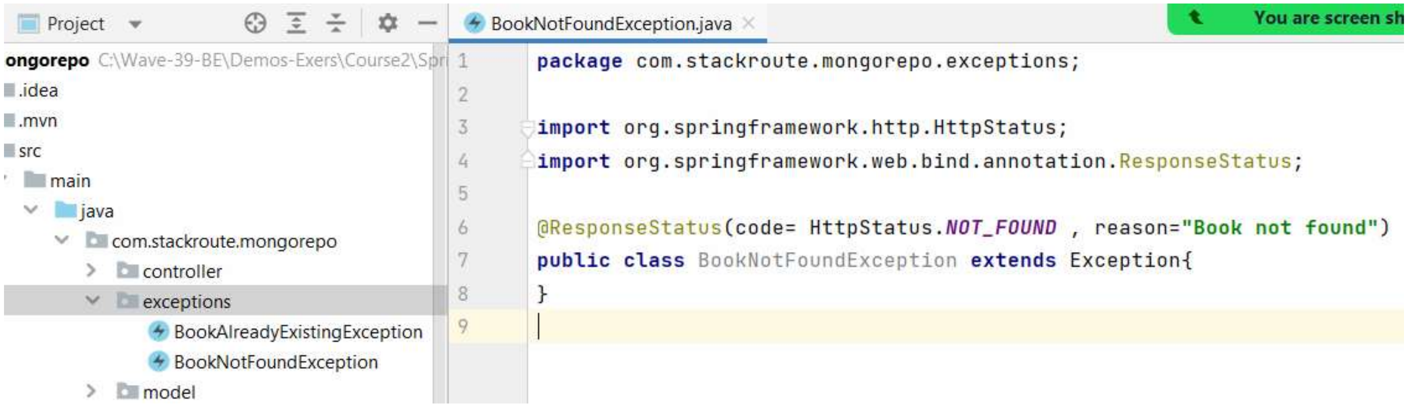
has to return record if found in db   message, httpstatus

else, throw exception

we need two custom exceptions


Project
BookAlreadyExistingException.java
You are screen sharing
ongorepo C:\Wave-39-BE\Demos-Exers\Course2\Spri
.idea
.mvn
src
main
java
com.stackroute.mongorepo
controller
exceptions
BookAlreadyExistingException
model

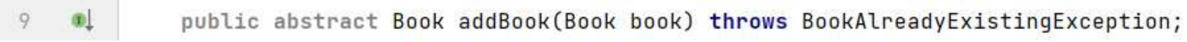
```java
package com.stackroute.mongorepo.exceptions;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(code= HttpStatus.CONFLICT, reason="Book already exists")
public class BookAlreadyExistingException extends Exception{
}
```



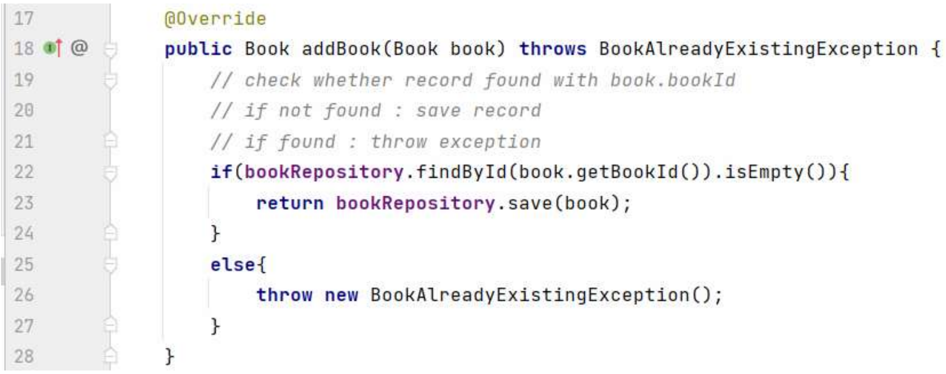
```java
package com.stackroute.mongorepo.exceptions;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(code= HttpStatus.NOT_FOUND , reason="Book not found")
public class BookNotFoundException extends Exception{
}
```

Task1 add exception in addbook()

Edit service method

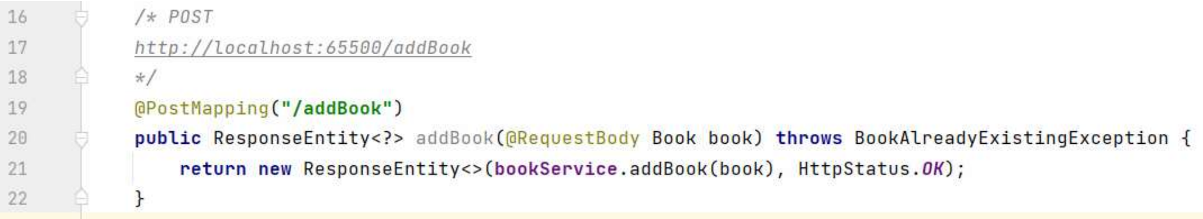


```java
public abstract Book addBook(Book book) throws BookAlreadyExistingException;
```



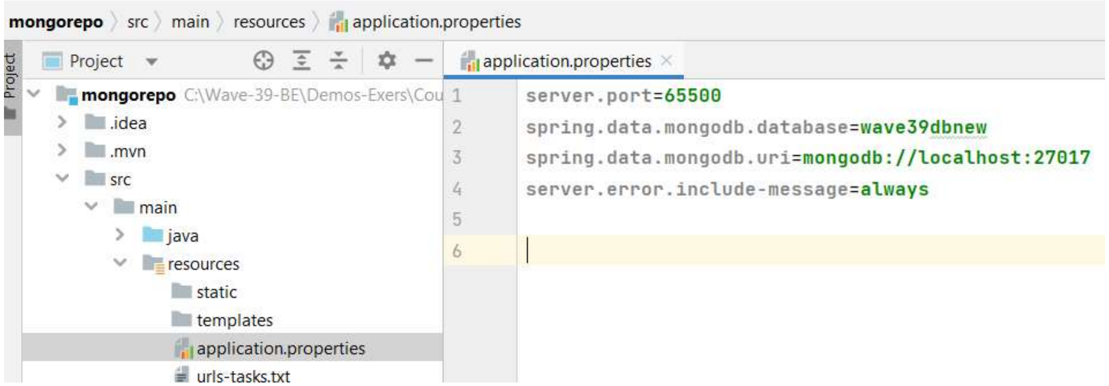
```java
@Override
public Book addBook(Book book) throws BookAlreadyExistingException {
    // check whether record found with book.bookId
    // if not found : save record
    // if found : throw exception
    if(bookRepository.findById(book.getBookId()).isEmpty()){
        return bookRepository.save(book);
    }
    else{
        throw new BookAlreadyExistingException();
    }
}
```

Edit controller

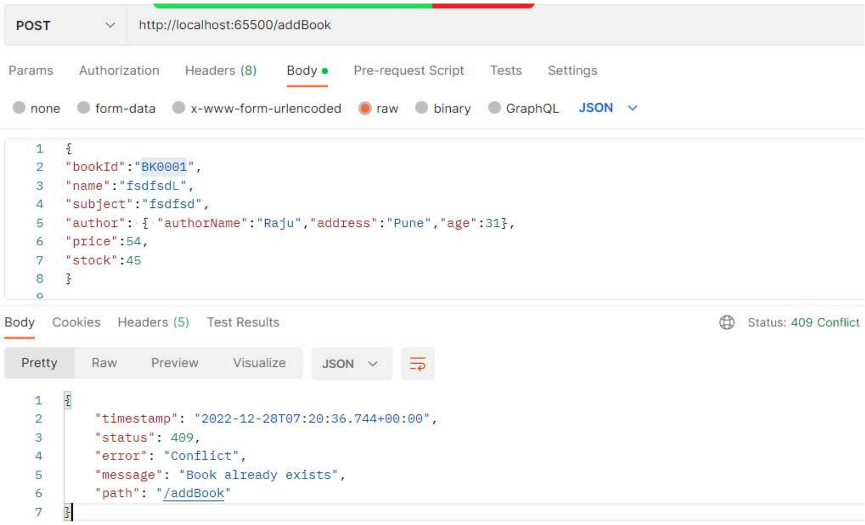


```
/* POST
http://localhost:65500/addBook
*/
@PostMapping("/addBook")
public ResponseEntity<?> addBook(@RequestBody Book book) throws BookAlreadyExistingException {
    return new ResponseEntity<>(bookService.addBook(book), HttpStatus.OK);
}
```

Edit application.properties



```
server.port=65500
spring.data.mongodb.database=wave39dbnew
spring.data.mongodb.uri=mongodb://localhost:27017
server.error.include-message=always

```

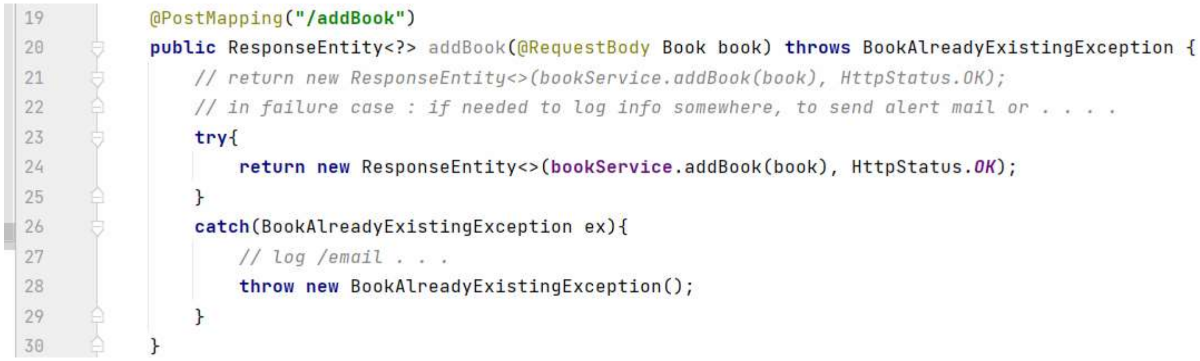


```
{
"bookId":"BK0001",
"name":"fsdfsdL",
"subject":"fsdfsd",
"author": { "authorName":"Raju","address":"Pune","age":31},
"price":54,
"stock":45
}
```

Status: 409 Conflict

```
{
    "timestamp": "2022-12-28T07:20:36.744+00:00",
    "status": 409,
    "error": "Conflict",
    "message": "Book already exists",
    "path": "/addBook"
}
```

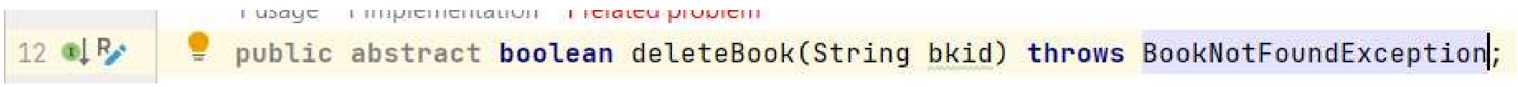
to get space to implement additional login in exception raised case



```
@PostMapping("/addBook")
public ResponseEntity<?> addBook(@RequestBody Book book) throws BookAlreadyExistingException {
    // return new ResponseEntity<>(bookService.addBook(book), HttpStatus.OK);
    // in failure case : if needed to log info somewhere, to send alert mail or . . . .
    try{
        return new ResponseEntity<>(bookService.addBook(book), HttpStatus.OK);
    }
    catch(BookAlreadyExistingException ex){
        // log /email . . .
        throw new BookAlreadyExistingException();
    }
}
```

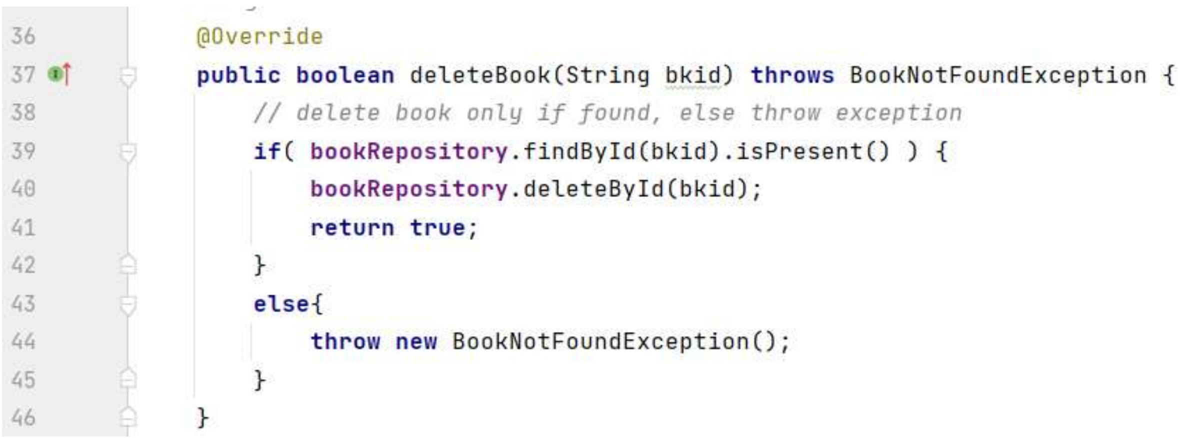
Task2 add exception in deleteBook()

Edit service method



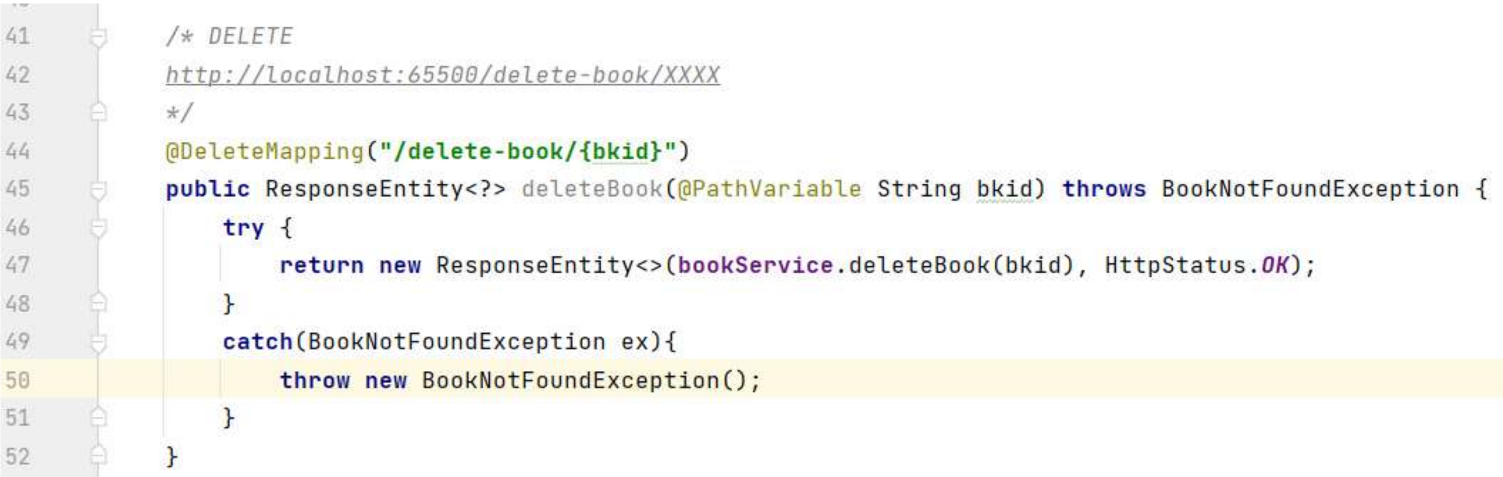
1 usage 1 implementation 1 related problem

```
public abstract boolean deleteBook(String bkid) throws BookNotFoundException;
```



```
@Override
public boolean deleteBook(String bkid) throws BookNotFoundException {
    // delete book only if found, else throw exception
    if( bookRepository.findById(bkid).isPresent() ) {
        bookRepository.deleteById(bkid);
        return true;
    }
    else{
        throw new BookNotFoundException();
    }
}
```

Edit controller method



```
/* DELETE
http://localhost:65500/delete-book/XXXX
*/
@DeleteMapping("/delete-book/{bkid}")
public ResponseEntity<?> deleteBook(@PathVariable String bkid) throws BookNotFoundException {
    try {
        return new ResponseEntity<>(bookService.deleteBook(bkid), HttpStatus.OK);
    }
    catch(BookNotFoundException ex){
        throw new BookNotFoundException();
    }
}
```

od