**Spring Boot Rest API**

Pagination API:

Entity:

**import** com.example.mouri.dto.BookDto;

**import** lombok.AllArgsConstructor;

**import** lombok.Data;

**import** lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

**public** **class** PostRespones {

**private** List<BookDto> content;

**private** **int** pageNo;

**private** **int** pageSize;

**private** **long** totalElements;

**private** **int** totalPages;

**private** **boolean** last;

}

Service :

PostRespones getAllBooks(**int** pageNo, **int** pageSize, String sortBy, String sortDir);

ServiceImpl:

@Override

**public** PostRespones getAllBooks(**int** pageNo, **int** pageSize, String sortBy, String sortDir) {

Sort sort = sortDir.equalsIgnoreCase(Sort.Direction.***ASC***.name()) ? Sort.*by*(sortBy).ascending():

Sort.*by*(sortBy).descending();

Pageable pageable = PageRequest.*of*(pageNo, pageSize, sort);

Page<Book> book = bookRepository.findAll(pageable);

List<Book> listBook = book.getContent();

List<BookDto> content = listBook.stream().map(b -> mappers.toBookDto(b)).collect(Collectors.*toList*());

PostRespones postRespones = **new** PostRespones();

postRespones.setContent(content);

postRespones.setPageNo(book.getNumber());

postRespones.setPageSize(book.getSize());

postRespones.setTotalElements(book.getTotalElements());

postRespones.setTotalPages(book.getTotalPages());

postRespones.setLast(book.isLast());

**return** postRespones;

}

Controller:

@GetMapping("/passengerList")

**public** PostRespones getAllBooks(

@RequestParam(value="pageNo", defaultValue="0",required = **false**) **int** pageNo,

@RequestParam(value="pageSize", defaultValue="10",required = **false**) **int** pageSize,

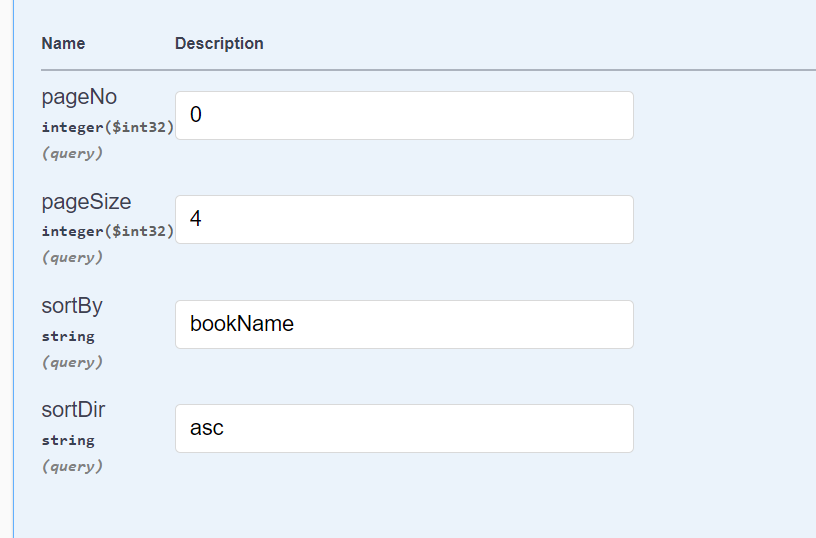
@RequestParam(value="sortBy", defaultValue="id",required = **false**) String sortBy,

@RequestParam(value="sortDir", defaultValue="asc",required = **false**) String sortDir){

**return** bookService.getAllBooks(pageNo,pageSize,sortBy,sortDir);

}

OutPut:



Search and Filter API:

Repository:

@Query("SELECT b FROM Book b WHERE " +

"b.bookAuthor LIKE CONCAT('%',:query,'%')" + " Or b.bookName LIKE CONCAT('%',:query,'%')")

List<Book> searchBook(String query);

Service:

List<BookDto> searchBook(String query);

ServiceImpl:

@Override

**public** List<BookDto> searchBook(String query) {

List<Book> book= bookRepository.searchBook(query);

**return** book.stream().map(b ->{

BookDto bookDto= **new** BookDto();

bookDto.setBookId(b.getBookId());

bookDto.setBookName(b.getBookName());

bookDto.setBookAuthor(b.getBookAuthor());

**return** bookDto;

}).collect(Collectors.*toList*());

}

Controller:

@GetMapping("/search")

**public** List<BookDto> searchBook(@RequestParam("query")String query) {

**return** bookService.searchBook(query);

}

OutPut:

