

Nama : Muh. Nur Afrizal

NIM : 222212738

Kelas / No : 3SI1 / 22

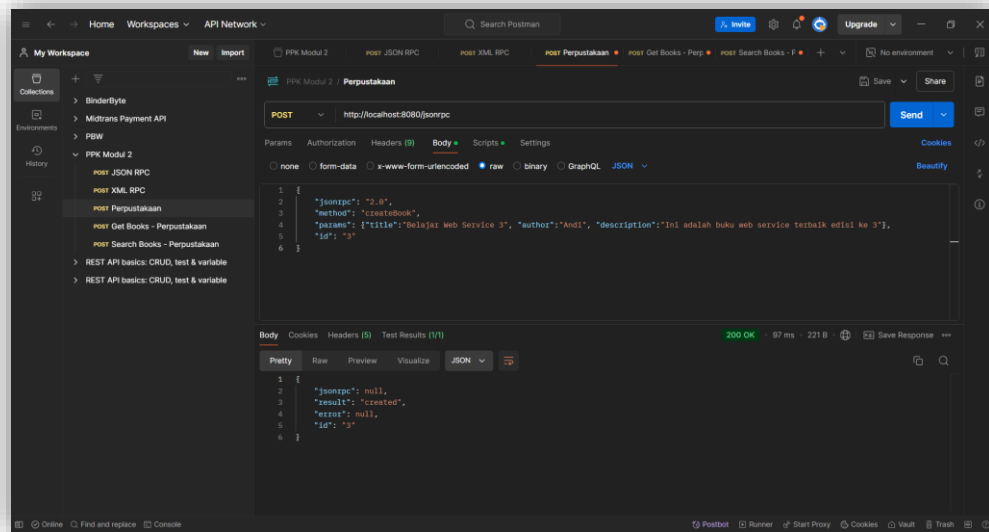
# LAPORAN PRAKTIKUM PEMROGRAMAN PLATFORM KHUSUS

## MODUL 2

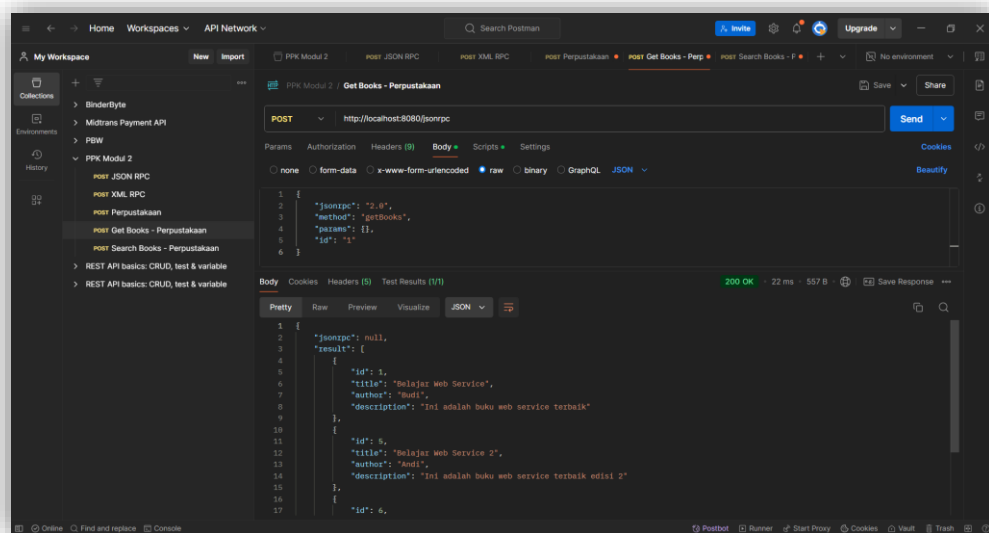
### JSON-RPC dan XML-RPC

#### A. Output

##### 1. Method createBook

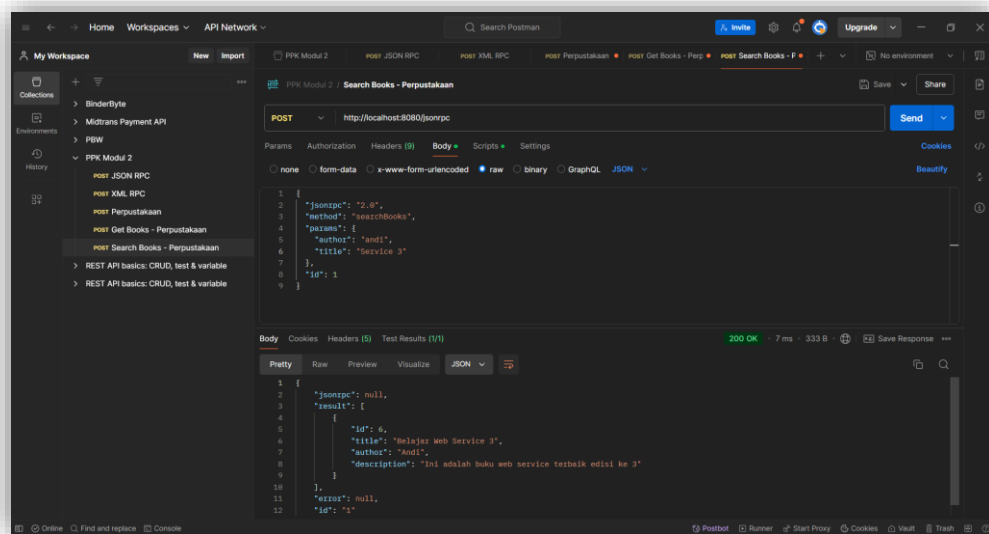
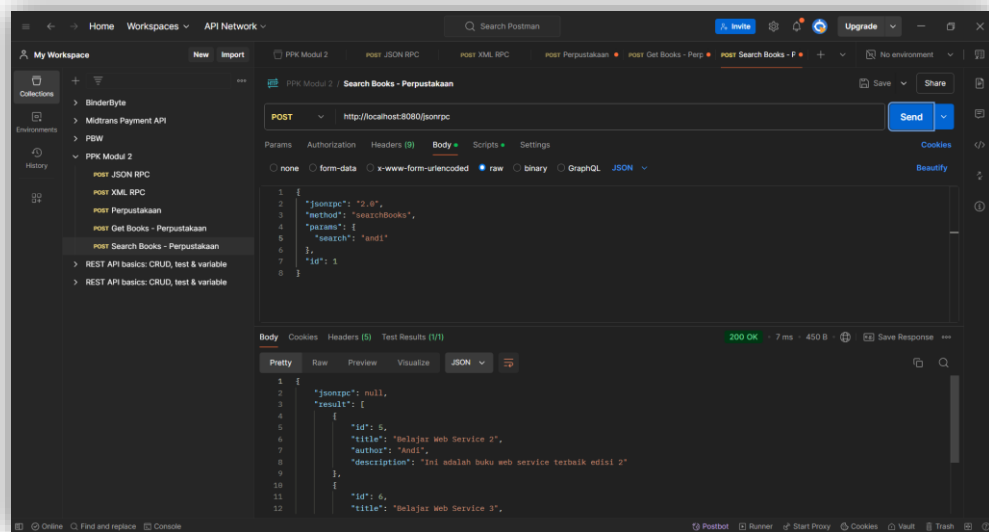


##### 2. Method getBooks



Nama : Muh. Nur Afrizal  
NIM : 222212738  
Kelas / No : 3SI1 / 22

### 3. Method searchBooks



Nama : Muh. Nur Afrizal

NIM : 222212738

Kelas / No : 3SI1 / 22

## B. JsonRpcController (Controller)

```
package com.polstat.perpustakaan.controller;

import ...

import java.util.List;

@RestController
public class JsonRpcController {
    @Autowired
    private BookService bookService;
    @PostMapping("/jsonrpc")
    public ResponseEntity<Object> handleJsonRpcRequest(@RequestBody JsonRpcRequest request) {
        try {
            String method = request.getMethod();
            JsonNode params = request.getParams();
            System.out.println("Method: " + method);
            switch (method) {
                case "createBook":
                    String title = params.get("title").asText();
                    String author = params.get("author").asText();
                    String description = params.get("description").asText();
                    BookDto book = BookDto.builder()
                        .title(title)
                        .description(description)
                        .author(author)
                        .build();
                    bookService.createBook(book);
                    return ResponseEntity.ok(new JsonRpcResponse("created", request.getId()));
                case "getBooks":
                    List<BookDto> books = bookService.getBooks();
                    return ResponseEntity.ok(new JsonRpcResponse(books, request.getId()));
                case "searchBooks":
                    if (params.has("author") && params.has("title")) {
                        author = params.get("author").asText();
                        title = params.get("title").asText();
                        List<BookDto> searchResults = bookService.searchBooks(author, title);
                        return ResponseEntity.ok(new JsonRpcResponse(searchResults, request.getId()));
                    } else if (params.has("search")) {
                        String search = params.get("search").asText();
                        List<BookDto> searchResults = bookService.searchBooks(search);
                        return ResponseEntity.ok(new JsonRpcResponse(searchResults, request.getId()));
                    }
                default:
                    return ResponseEntity.badRequest().build();
            }
        } catch (Exception e) {
            return ResponseEntity.badRequest().build();
        }
    }
}
```

- **@PostMapping("/jsonrpc")**: Mendefinisikan endpoint POST di /jsonrpc.
  - a. **createBook**:
    - Mengekstrak title, author, dan description dari params.
    - Membuat BookDto dan menyimpannya menggunakan bookService.
    - Mengembalikan respons "created".
  - b. **getBooks**:
    - Memanggil bookService.getBooks() untuk mendapatkan semua buku.

Nama : Muh. Nur Afrizal

NIM : 222212738

Kelas / No : 3SI1 / 22

- Mengembalikan list buku dalam respons.
- c. `searchBooks`:
  - Jika `author` dan `title` ada memanggil `bookService.searchBooks(author, title)`.
  - Jika hanya `search` yang ada memanggil `bookService.searchBooks(search)`.
  - Mengembalikan hasil pencarian dalam respons.
- Error Handling:
  - Menggunakan try-catch untuk menangkap exception.
  - Mengembalikan `ResponseEntity.badRequest()` jika terjadi error atau metode tidak dikenali.
- Respons:
  - Menggunakan `JsonRpcResponse` untuk membungkus hasil dan ID request.
  - Mengembalikan `ResponseEntity.ok()` dengan respons yang sesuai.

### C. BookDto (DTO)

```
package com.polstat.perpustakaan.dto;

import ...

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class BookDto {
    private Long id;

    @NotEmpty(message = "Judul buku wajib diisi.")
    private String title;
    @NotNull(message = "Penulis buku wajib diisi.")
    private String author;
    private String description;
}
```

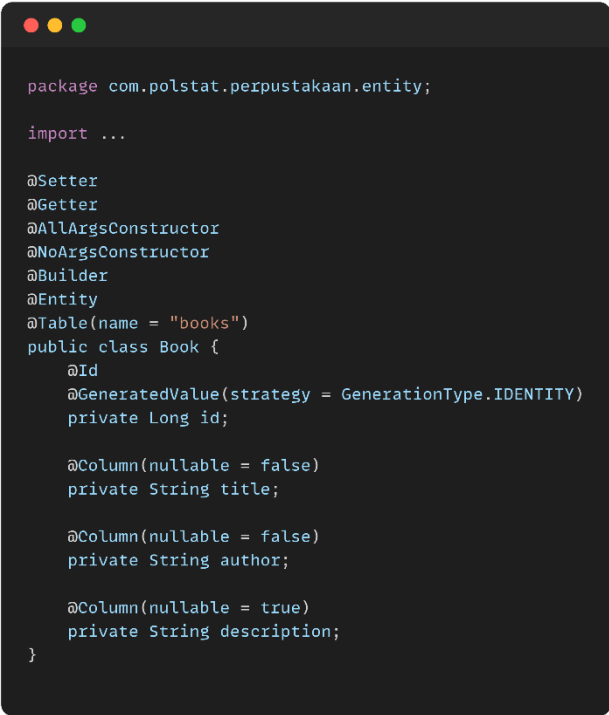
Kelas ini digunakan untuk mentransfer data buku dalam aplikasi, misalnya antara controller dan service, atau ketika mengirim/menerima data buku melalui API. Validasi yang ditambahkan memastikan integritas data sebelum diproses lebih lanjut.

Nama : Muh. Nur Afrizal

NIM : 222212738

Kelas / No : 3SI1 / 22

#### D. Book (Entity)

A screenshot of a code editor with a dark background and light-colored text. The code is in Java and defines a JPA entity class named 'Book'. It includes package and import statements, followed by a series of annotations: @Setter, @Getter, @AllArgsConstructor, @NoArgsConstructor, @Builder, @Entity, and @Table(name = "books"). The class itself is public and contains three private fields: 'id' (Long), 'title' (String), and 'author' (String), each with its own annotation. The 'description' field is also private (String) but is not annotated. The code is enclosed in curly braces.

```
package com.polstat.perpustakaan.entity;

import ...

@Setter
@Getter
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Entity
@Table(name = "books")
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)
    private String title;

    @Column(nullable = false)
    private String author;

    @Column(nullable = true)
    private String description;
}
```

Entitas Book ini mendefinisikan struktur data buku yang akan disimpan dalam database. Ini adalah representasi objek dari tabel 'books' dalam konteks JPA (Java Persistence API). Anotasi-anotasi yang digunakan memungkinkan ORM (Object-Relational Mapping) untuk mengelola persistensi data antara objek Java dan database relasional.

Nama : Muh. Nur Afrizal

NIM : 222212738

Kelas / No : 3SI1 / 22

## E. BookMapper (Mapper)

```
package com.polstat.perpustakaan.mapper;

import ...

public class BookMapper {
    public static Book mapToBook(BookDto bookDto){
        return Book.builder()
            .id(bookDto.getId())
            .title(bookDto.getTitle())
            .description(bookDto.getDescription())
            .author(bookDto.getAuthor())
            .build();
    }

    public static BookDto mapToBookDto(Book book){
        return BookDto.builder()
            .id(book.getId())
            .title(book.getTitle())
            .description(book.getDescription())
            .author(book.getAuthor())
            .build();
    }
}
```

BookMapper digunakan untuk mentransformasi data buku antara format yang digunakan di frontend (DTO) dan format yang digunakan untuk penyimpanan di database (Entity).

## F. BookRepository (Repository)

```
package com.polstat.perpustakaan.repository;

import ...

import java.util.List;

public interface BookRepository extends JpaRepository<Book, Long> {
    @Query("SELECT b FROM Book b WHERE LOWER(b.title) LIKE LOWER(CONCAT('%', :searchTerm, '%')) OR LOWER(b.author) LIKE LOWER(CONCAT('%', :searchTerm, '%'))")
    List<Book> searchBooks(@Param("searchTerm") String searchTerm);

    @Query("SELECT b FROM Book b WHERE LOWER(b.author) LIKE LOWER(CONCAT('%', :author, '%')) AND LOWER(b.title) LIKE LOWER(CONCAT('%', :title, '%'))")
    List<Book> searchBooksByAuthorAndTitle(@Param("author") String author, @Param("title") String title);
}
```

BookRepository yaitu antarmuka yang menangani operasi database untuk buku, termasuk pencarian umum dan pencarian spesifik berdasarkan penulis dan judul.

Nama : Muh. Nur Afrizal

NIM : 222212738

Kelas / No : 3SI1 / 22

## G. BooService (Service)

```
package com.polstat.perpustakaan.service;

import ...

@Service
public class BookServiceImpl implements BookService {
    @Autowired
    private BookRepository bookRepository;

    @Override
    public void createBook(BookDto bookDto) {
        bookRepository.save(BookMapper.mapToBook(bookDto));
    }
    @Override
    public List<BookDto> getBooks() {
        List<Book> books = bookRepository.findAll();
        List<BookDto> bookDtos = books.stream()
            .map((product) -> (BookMapper.mapToBookDto(product)))
            .collect(Collectors.toList());
        return bookDtos;
    }

    public List<BookDto> searchBooks(String searchTerm) {
        return bookRepository.searchBooks(searchTerm).stream()
            .map(BookMapper::mapToBookDto)
            .collect(Collectors.toList());
    }

    @Override
    public List<BookDto> searchBooks(String author, String title) {
        return bookRepository.searchBooksByAuthorAndTitle(author, title).stream()
            .map(BookMapper::mapToBookDto)
            .collect(Collectors.toList());
    }
}
```

BookServiceImpl menangani logika bisnis terkait operasi buku, bertindak sebagai perantara antara controller dan repository, serta menangani konversi data antara entitas dan DTO.

Nama : Muh. Nur Afrizal

NIM : 222212738

Kelas / No : 3SI1 / 22