3.6 Penugasan

Ubahlah layanan perpustakaan pada modul praktikum 2 ke versi SOAP Web Service. Layanan yang dibuat adalah menambahkan koleksi buku dan mendapatkan semua koleksi buku. Lakukan pengujian untuk setiap layanan yang dibuat. Selanjutnya, buat laporan pekerjaan Anda dengan penjelasan lengkap dalam format pdf.

Jawab :

Untuk mengonversi layanan perpustakaan sebelumnya ke versi SOAP, terdapat beberapa hal yang perlu ditambahkan, sedangkan sisanya tidak perlu ditambah. Hal – hal yang perlu ditambah adalah sebagai berikut :

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.3.4</version>
        <relativePath /> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.polstat</groupId>
    <artifactId>perpustakaan</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>perpustakaan</name>
    <description>Layanan perpustakaan</description>
    <url />
    <licenses>
        <license />
    </licenses>
    <developers>
        <developer />
    </developers>
    <scm>
        <connection />
        <developerConnection />
        <tag />
```

```
        <url />
    </scm>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>com.mysql</groupId>
            <artifactId>mysql-connector-j</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>jakarta.validation</groupId>
            <artifactId>jakarta.validation-api</artifactId>
            <version>3.0.2</version>
            <type>jar</type>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web-services</artifactId>
        </dependency>
        <dependency>
            <groupId>jakarta.xml.bind</groupId>
            <artifactId>jakarta.xml.bind-api</artifactId>
            <version>4.0.0</version>
```

```
        </dependency>
        <dependency>
            <groupId>org.glassfish.jaxb</groupId>
            <artifactId>jaxb-runtime</artifactId>
            <version>3.0.2</version>
        </dependency>
        <dependency>
            <groupId>wsdl4j</groupId>
            <artifactId>wsdl4j</artifactId>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>jaxb2-maven-plugin</artifactId>
                <version>3.1.0</version>
                <executions>
                    <execution>
                        <id>xjc</id>
                        <goals>
                            <goal>xjc</goal>
                        </goals>
                    </execution>
                </executions>
                <configuration>
                    <sources>
                        <source>src/main/resources/perpustakaan.xsd</source>
                    </sources>
                    <outputDirectory>src/main/java</outputDirectory>
                    <clearOutputDir>false</clearOutputDir>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

File pom.xml ini mengatur proyek Spring Boot yang berinteraksi dengan database, menerapkan JPA untuk persistensi data, mendukung layanan web (baik REST maupun SOAP), dan memanfaatkan JAXB untuk pengikatan XML.

application.properties

```
spring.application.name=perpustakaan
spring.datasource.url=jdbc:mysql://localhost:3306/perpustakaan
spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect
```

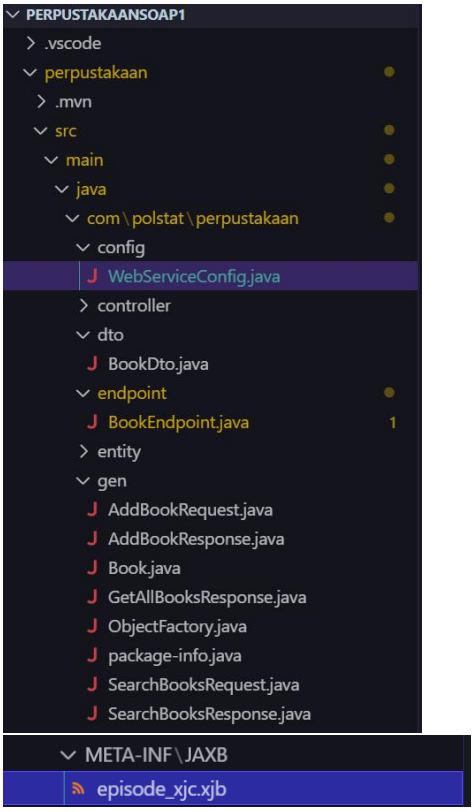
Application.properties ini digunakan untuk menghubungkan sistem dengan database di phpMyAdmin.

perpustakaan.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://www.polstat.com/perpustakaan"
targetNamespace="http://www.polstat.com/perpustakaan"
elementFormDefault="qualified">

    <xs:element name="addBookRequest">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="title" type="xs:string" />
                <xs:element name="author" type="xs:string" />
                <xs:element name="description" type="xs:string" minOccurs="0" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="addBookResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="status" type="xs:string" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="getAllBooksRequest">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="books" type="xs:string" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
```

```
    <xs:element name="getAllBooksResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="books" type="tns:book" maxOccurs="unbounded" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="searchBooksRequest">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="keyword" type="xs:string" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="searchBooksResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="books" type="tns:book" maxOccurs="unbounded" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="book">
        <xs:sequence>
            <xs:element name="id" type="xs:long" />
            <xs:element name="title" type="xs:string" />
            <xs:element name="author" type="xs:string" />
            <xs:element name="description" type="xs:string" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>

</xs:schema>
```

File perpustakaan.xsd ini dibuat untuk mendefinisikan struktur dan format dari pesan yang akan digunakan dalam layanan web (web service) perpustakaan. Setelah ini, proyek akan di*build with dependencies* dan akan memunculkan file yang di*generate* secara otomatis dengan directory yang telah disesuaikan dengan outputDir di pom.xml sebelumnya. File yang di*generate* secara otomatis diantaranya adalah sebagai berikut :



WebServiceConfig.java

```java
package com.polstat.perpustakaan.config;

import org.springframework.boot.web.servlet.ServletRegistrationBean;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.io.ClassPathResource;
import org.springframework.ws.config.annotation.EnableWs;
import org.springframework.ws.transport.http.MessageDispatcherServlet;
import org.springframework.ws.wsdl.wsdl11.DefaultWsdl11Definition;
import org.springframework.xml.xsd.SimpleXsdSchema;
import org.springframework.xml.xsd.XsdSchema;

@EnableWs
@Configuration
public class WebServiceConfig {
```

```
    @Bean
    public ServletRegistrationBean<MessageDispatcherServlet>
messageDispatcherServlet(ApplicationContext context) {
        MessageDispatcherServlet servlet = new MessageDispatcherServlet();
        servlet.setApplicationContext(context);
        servlet.setTransformWsdlLocations(true);
        return new ServletRegistrationBean<>(servlet, "/ws/*");
    }

    @Bean(name = "perpustakaan")
    public DefaultWsdl11Definition defaultWsdl11Definition(XsdSchema
librarySchema) {
        DefaultWsdl11Definition definition = new DefaultWsdl11Definition();
        definition.setPortTypeName("LibraryPort");
        definition.setLocationUri("/ws");
        definition.setTargetNamespace("http://www.polstat.com/perpustakaan");
        definition.setSchema(librarySchema);
        return definition;
    }

    @Bean
    public XsdSchema librarySchema() {
        return new SimpleXsdSchema(new ClassPathResource("perpustakaan.xsd"));
    }
}
```

File ini berfungsi sebagai konfigurasi untuk layanan web menggunakan Spring Web Services. File ini memungkinkan *user* untuk berinteraksi dengan layanan perpustakaan melalui protokol SOAP dengan format yang sesuai dan *path* bernama "/ws".

BookEndpoint.java

```
package com.polstat.perpustakaan.endpoint;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import com.polstat.perpustakaan.dto.BookDto;
import com.polstat.perpustakaan.service.BookService;
import com.polstat.perpustakaan.gen.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.ws.server.endpoint.annotation.Endpoint;
import org.springframework.ws.server.endpoint.annotation.PayloadRoot;
import org.springframework.ws.server.endpoint.annotation.RequestPayload;
import org.springframework.ws.server.endpoint.annotation.ResponsePayload;
```

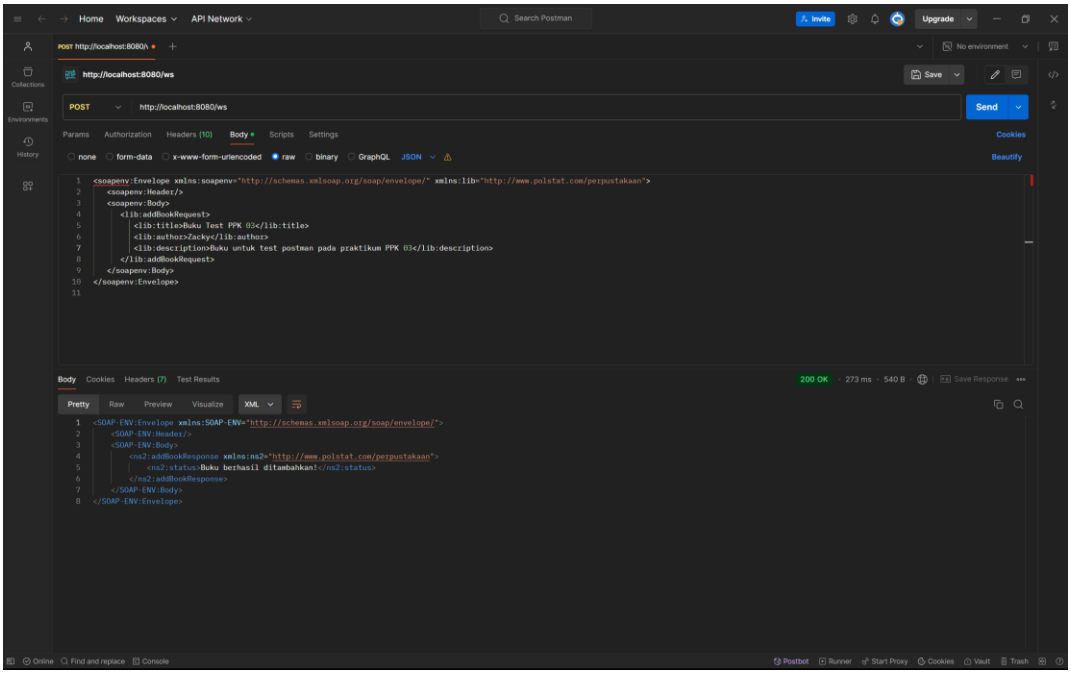
```java
import java.util.List;

@Endpoint
public class BookEndpoint {
    private static final Logger logger =
LoggerFactory.getLogger(BookEndpoint.class);

    private static final String NAMESPACE_URI =
"http://www.polstat.com/perpustakaan";

    @Autowired
    private BookService bookService;

    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "getAllBooksRequest")
    @ResponsePayload
    public GetAllBooksResponse getBooks() {
        GetAllBooksResponse response = new GetAllBooksResponse();

        List<BookDto> books = bookService.getBooks();
        for (BookDto book : books) {
            Book book2 = new Book();
            book2.setId(book.getId());
            book2.setTitle(book.getTitle());
            book2.setAuthor(book.getAuthor());
            book2.setDescription(book.getDescription());

            response.getBooks().add(book2);
        }

        return response;
    }

    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "addBookRequest")
    @ResponsePayload
    public AddBookResponse addBook(@RequestPayload AddBookRequest request) {
        AddBookResponse response = new AddBookResponse();
        BookDto bookDto = new BookDto(null, request.getTitle(),
request.getAuthor(), request.getDescription());
        bookService.createBook(bookDto);
        response.setStatus("Buku berhasil ditambahkan!");
        return response;
    }

    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "searchBooksRequest")
```

```
    @ResponsePayload
    public SearchBooksResponse searchBooks(@RequestPayload SearchBooksRequest
request) {
        SearchBooksResponse response = new SearchBooksResponse();
        List<BookDto> books = bookService.searchBooks(request.getKeyword());
        for (BookDto bookDto : books) {
            response.getBooks().add(mapToBook(bookDto));
        }
        return response;
    }

    private Book mapToBook(BookDto bookDto) {
        Book book = new Book();
        book.setId(bookDto.getId());
        book.setTitle(bookDto.getTitle());
        book.setAuthor(bookDto.getAuthor());
        book.setDescription(bookDto.getDescription());
        return book;
    }
}
```

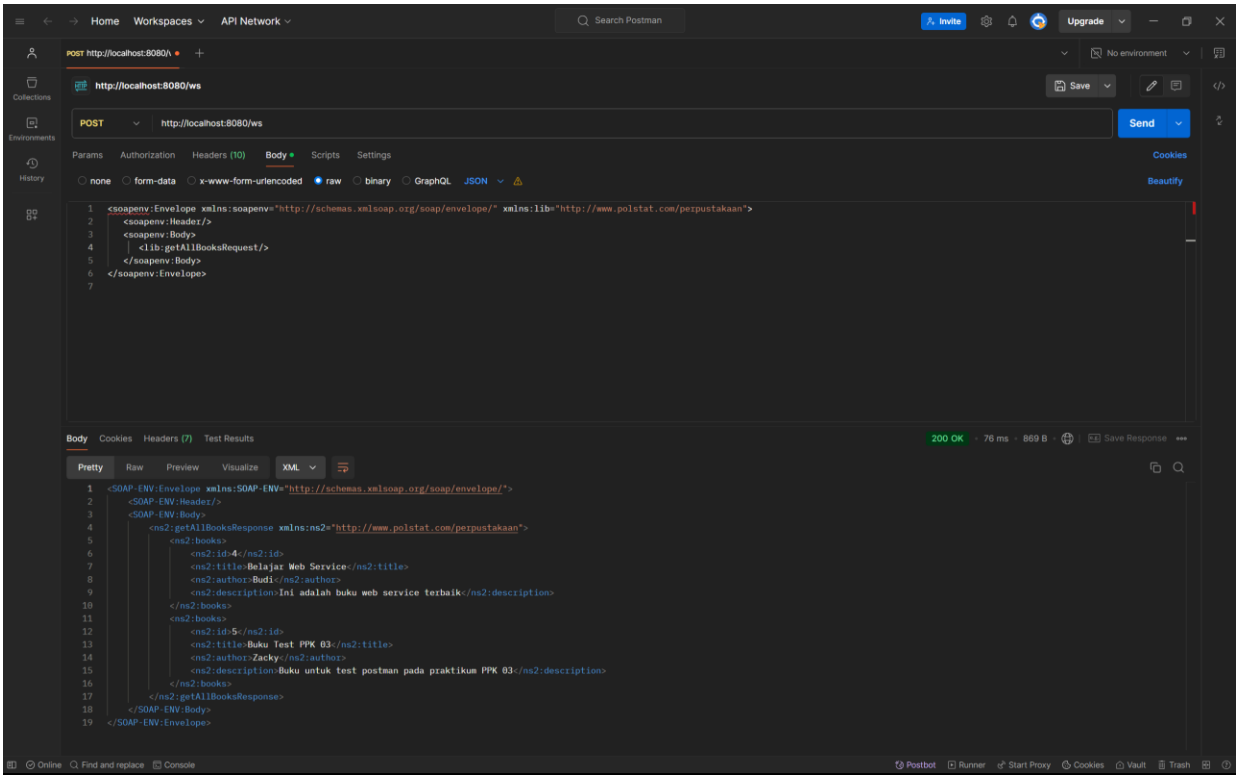
File di atas mendefinisikan endpoint untuk layanan web perpustakaan menggunakan Spring Web Services. Endpoint ini menangani permintaan untuk mendapatkan semua buku, menambahkan buku baru, dan mencari buku berdasarkan kata kunci, serta mengonversi data dari dan ke objek DTO yang sesuai.

Hasil Pengujian :

1. Uji untuk tambah / add buku :



2. Uji untuk mendapatkan / get buku :



3. Uji untuk mencari buku berdasarkan keyword :