MODUL 3 Simple Object Access Protocol (SOAP)

3.1 Deskripsi Singkat

SOAP (Simple Object Access Protocol) adalah protokol standar untuk pertukaran pesan di lingkungan web service. Dirancang untuk mendukung komunikasi antara aplikasi yang berjalan di platform yang berbeda dan bahasa pemrograman yang beragam, SOAP memungkinkan data dan fungsi dipertukarkan dengan aman melalui jaringan. SOAP menggunakan format pesan yang independen dari platform, berfokus pada layanan yang dapat diandalkan dan keamanan dalam komunikasi.

Pesan SOAP adalah pusat dari komunikasi SOAP. Setiap pesan SOAP mengandung elemen-elemen seperti Envelope, Header, dan Body. Envelope adalah kontainer luar yang mengelilingi pesan SOAP, sementara Header berisi informasi opsional yang relevan dengan pesan, seperti keamanan atau routing. Bagian paling penting adalah Body, di mana isi pesan sebenarnya ditempatkan. Format pesan SOAP menggunakan XML, yang memungkinkan aplikasi untuk menguraikan dan memproses pesan dengan mudah.

SOAP menawarkan keamanan yang baik dengan dukungan untuk berbagai protokol keamanan seperti HTTPS dan WS-Security. Ini juga mendukung kemampuan untuk mengatasi kesalahan dan pemulihan yang terjadi selama komunikasi. Keuntungan lainnya adalah keserbagunaan - aplikasi yang ditulis dalam bahasa pemrograman yang berbeda dapat berkomunikasi menggunakan SOAP. SOAP umumnya digunakan dalam lingkungan perusahaan yang memerlukan keandalan dan keamanan yang tinggi, seperti integrasi antara sistem yang berbeda, arsitektur berorientasi layanan (SOA), serta saat membutuhkan kontrol yang lebih tinggi atas pesan dan operasi yang dilakukan.

Kritik terhadap SOAP sering terkait dengan kompleksitas dan overheadnya. Format XML yang digunakan dalam SOAP cenderung lebih berat dan sulit dibaca oleh manusia dibandingkan dengan format data lain seperti JSON. Selain itu, pengaturan dan konfigurasi keamanan dalam SOAP dapat menjadi rumit. Pada saat ini, banyak pengembang lebih

memilih protokol lain seperti REST dan JSON untuk komunikasi web service karena kesederhanaan, efisiensi, dan ketersediaan dukungan yang lebih luas.

SOAP tetap menjadi salah satu protokol utama dalam lingkungan web service, terutama dalam kasus di mana keamanan, keandalan, dan fitur dukungan yang kuat lebih penting daripada efisiensi. Meskipun kompleksitasnya mungkin menghalangi beberapa pengembang, SOAP tetap menjadi pilihan yang baik untuk integrasi bisnis yang memerlukan standar komunikasi dan kontrol yang tinggi. Dengan melibatkan XML sebagai format pesan, SOAP tetap relevan dalam lingkungan yang berfokus pada interaksi yang terstruktur dan berorientasi pada layanan.

3.2 Tujuan Praktikum

Tujuan praktikum ini adalah membuat suatu layanan menggunakan SOAP yang diimplementasikan dengan Spring Framework.

Setelah praktikum ini kompetensi yang akan dicapai adalah mahasiswa mampu memahami konsep teknologi web service dan memahami implementasi web service dengan SOAP.

3.3 Alokasi Waktu

Alokasi waktu pada praktikum 1 adalah sebagai berikut :

1. Tatap muka di kelas : 50 menit

2. Kerja mandiri (mengerjakan penugasan) : 120 menit

3.4 Material Praktikum

Praktikum ini memerlukan beberapa material sebagai berikut:

1) Java Development Kit (JDK) versi 17,

2) Java IDE: Netbeans, IntelliJ IDEA, atau Spring Tool Suite (STS),

3) Maven 3.5+,

4) Postman https://www.postman.com/downloads/

3.5 Kegiatan Praktikum

Pada praktikum ini, kita akan membuat sebuah layanan dengan SOAP Web Service untuk mendapatkan data suatu negara berdasarkan nama. Proyek dibuat menggunakan Spring Boot dan beberapa library pendukung. Ikuti langkah-langkah berikut ini.

Langkah 1: Menyiapkan Proyek

1) Buat proyek Spring Boot melalui NetBeans IDE atau IDE lainnya. Klik tombol New Project. Pilih Java with Maven dan Java Application. Isikan nama proyek **soap**.

2) Tambahkan dependency pada file pom.xml. Buka file pom.xml dan ubah seperti berikut.

```
<?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
http://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>3.1.2</version>
    <relativePath/>
  </parent>
  <groupId>com.example</groupId>
  <artifactId>soap</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web-services</artifactId>
    </dependency>
    <dependency>
```

```
        <groupId>wsdl4j</groupId>
        <artifactId>wsdl4j</artifactId>
      </dependency>
      <dependency>
        <groupId>jakarta.xml.bind</groupId>
        <artifactId>jakarta.xml.bind-api</artifactId>
        <version>4.0.0</version>
      </dependency>
      <dependency>
        <groupId>javax.xml.bind</groupId>
        <artifactId>jaxb-api</artifactId>
        <version>2.3.1</version>
      </dependency>
      <dependency>
        <groupId>org.glassfish.jaxb</groupId>
        <artifactId>jaxb-runtime</artifactId>
      </dependency>
    </dependencies>
</project>
```

3) Klik menu *Build with Dependencies* agar *dependency library* yang dibutuhkan dapat diunduh. Tunggu sampai prosesnya selesai.

Langkah 2: Membuat Kode Program

1) Membuat XSD File

Domain web service didefinisikan dalam XML schema file (XSD). Spring-WS akan otomatis mengekspornya sebagai WSDL.

Buat file XSD yang di dalamnya terdapat operasi untuk mengembalikan nama negara, populasi, ibukota dan matauang. Berikut kode XSD file *countries.xsd* yang dibuat pada folder *src/main/resources*.

src/main/resources/countries.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://www.example.com/soap/gen"
        targetNamespace="http://www.example.com/soap/gen"
elementFormDefault="qualified">

    <xs:element name="getCountryRequest">
      <xs:complexType>
```

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

    <xs:element name="getCountryResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="country" type="tns:country"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>

    <xs:complexType name="country">
      <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="population" type="xs:int"/>
        <xs:element name="capital" type="xs:string"/>
        <xs:element name="currency" type="tns:currency"/>
      </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="currency">
      <xs:restriction base="xs:string">
        <xs:enumeration value="GBP"/>
        <xs:enumeration value="EUR"/>
        <xs:enumeration value="PLN"/>
      </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

2) Men-*generate* Kelas Domain

Domain yang didefinisikan sebelumnya akan di-generate otomatis menggunakan librabry jaxb2-maven-plugin.

Tambahkan kode berikut ke dalam file pom.xml *(setelah penutup tag </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/countries.xsd</source>
              </sources>
              <outputDirectory>src/main/java</outputDirectory>
              <clearOutputDir>false</clearOutputDir>
            </configuration>
          </plugin>
        </plugins>
      </build>
```

Klik menu *Build with Dependencies* pada proyek untuk men-*generate* kelas domain. Setelah itu akan ter-*generate* beberapa file di dalam package *com.example.soap.gen.*

3) Membuat Kelas Country Repository

Kelas ini menyediakan method untuk mendapatkan negara berdasarkan nama negara.

Buatlah file CountryRepository.java pada package com.example.soap. Berikut kode program selengkapnya.

```
package com.example.soap;

import com.example.soap.gen.Country;
import com.example.soap.gen.Currency;
import jakarta.annotation.PostConstruct;
import java.util.HashMap;
import java.util.Map;

import org.springframework.stereotype.Component;
import org.springframework.util.Assert;

@Component
public class CountryRepository {

   private static final Map<String, Country> countries = new HashMap<>();

   @PostConstruct
   public void initData() {
      Country spain = new Country();
      spain.setName("Spain");
      spain.setCapital("Madrid");
```

```
        spain.setCurrency(Currency.EUR);
        spain.setPopulation(46704314);

        countries.put(spain.getName(), spain);

        Country poland = new Country();
        poland.setName("Poland");
        poland.setCapital("Warsaw");
        poland.setCurrency(Currency.PLN);
        poland.setPopulation(38186860);

        countries.put(poland.getName(), poland);

        Country uk = new Country();
        uk.setName("United Kingdom");
        uk.setCapital("London");
        uk.setCurrency(Currency.GBP);
        uk.setPopulation(63705000);

        countries.put(uk.getName(), uk);
    }

    public Country findCountry(String name) {
        Assert.notNull(name, "The country's name must not be null");
        return countries.get(name);
    }
}
```

4) Membuat SOAP Endpoint

Kelas SOAP Endpoint akan menangani request SOAP, memprosesnya, dan mengirim response balik.

Buatlah file CountyEndpoint.java pada package com.example.soap. Berikut kode program selengkapnya.

```
package com.example.soap;

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;

import com.example.soap.gen.GetCountryRequest;
import com.example.soap.gen.GetCountryResponse;

@Endpoint
```

```
public class CountryEndpoint {

   private static final String NAMESPACE_URI = "http://example.com/soap/gen";

   private CountryRepository countryRepository;

   @Autowired
   public CountryEndpoint(CountryRepository countryRepository) {
      this.countryRepository = countryRepository;
   }

   @PayloadRoot(namespace = NAMESPACE_URI, localPart = "getCountryRequest")
   @ResponsePayload
   public GetCountryResponse getCountry(@RequestPayload GetCountryRequest
request) {
      GetCountryResponse response = new GetCountryResponse();
      response.setCountry(countryRepository.findCountry(request.getName()));

      return response;
   }
}
```

Penjelasan:

- *@Endpoint* – mendaftarkan kelas pada Spring WS sebagai Web Service Endpoint
- *@PayloadRoot* – mendefinisikan metode penanganan sesuai dengan atribut-atribut namespace dan localPart.
- *@ResponsePayload* – mengindikasikan bahwa metode ini mengembalikan suatu nilai yang akan dipetakan ke response payload.
- *@RequestPayload* – mengindikasikan bahwa metode ini menerima suatu parameter yang akan dipetakan dari permintaan yang masuk.

5) Membuat Kelas Konfigurasi SOAP Web Service

Kelas ini dibuat untuk mengkonfigurasi servlet pesan Spring agar dapat menerima request.

Buatlah file WebServiceConfig.java pada package com.example.soap. Berikut kode program selengkapnya.

```
package com.example.soap;

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.config.annotation.WsConfigurerAdapter;
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 extends WsConfigurerAdapter {

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

   @Bean(name = "countries")
   public DefaultWsdl11Definition defaultWsdl11Definition(XsdSchema
countriesSchema) {
      DefaultWsdl11Definition wsdl11Definition = new DefaultWsdl11Definition();
      wsdl11Definition.setPortTypeName("CountriesPort");
      wsdl11Definition.setLocationUri("/ws");
      wsdl11Definition.setTargetNamespace("http://www.example.com/soap/gen");
      wsdl11Definition.setSchema(countriesSchema);
      return wsdl11Definition;
   }

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

Kelas *WebServiceConfig* ini bertindak sebagai konfigurasi inti untuk mengaktifkan, mengkonfigurasi, dan menghubungkan komponen-

komponen yang diperlukan untuk mengimplementasikan layanan web SOAP menggunakan Spring Web Services.

- Kelas WebServiceConfig: kelas konfigurasi yang meng-extend WsConfigurerAdapter, yang memungkinkan kita untuk melakukan konfigurasi khusus untuk layanan web SOAP.
- *@EnableWs:* Ini adalah anotasi yang menandakan bahwa kelas ini akan mengaktifkan dukungan untuk layanan web SOAP dengan menggunakan Spring Web Services.
- *@Configuration:* Ini menunjukkan bahwa kelas ini adalah kelas konfigurasi Spring.
- *messageDispatcherServlet Bean:* Method ini mendefinisikan dan mengkonfigurasi servlet pengirim pesan Spring, yang bertanggung jawab untuk menangani permintaan yang masuk terkait layanan web. Ini mengatur transformasi lokasi WSDL dan mendaftarkan servlet dengan URL pemetaan "/ws/*".
- *defaultWsdl11Definition Bean:* Method ini mendefinisikan dan mengkonfigurasi definisi WSDL default. Ini menentukan port type name, location URI, target namespace, dan skema XML yang akan digunakan dalam WSDL.
- *countriesSchema Bean:* Method ini membuat dan mengkonfigurasi skema XSD yang akan digunakan untuk mendefinisikan struktur data yang akan ditukar dalam layanan web.

Langkah 3: Menguji Proyek SOAP

1) Menjalankan Proyek

Buat kelas Application sebagai executable class. Berikut kode program selengkapnya.

```
package com.example.soap;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
```

```
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Jalankan proyek dengan klik menu Run File pada file Application.java.

2) Menguji SOAP

Setelah aplikasi berhasil dijalankan, lakukan pengujian menggunakan Postman.

a) Buat request baru di postman dengan url http://localhost:8080/ws dan method POST.

b) Atur request header di tab Header. Tambahkan key: *Content-Type* dengan value: *text/xml.*

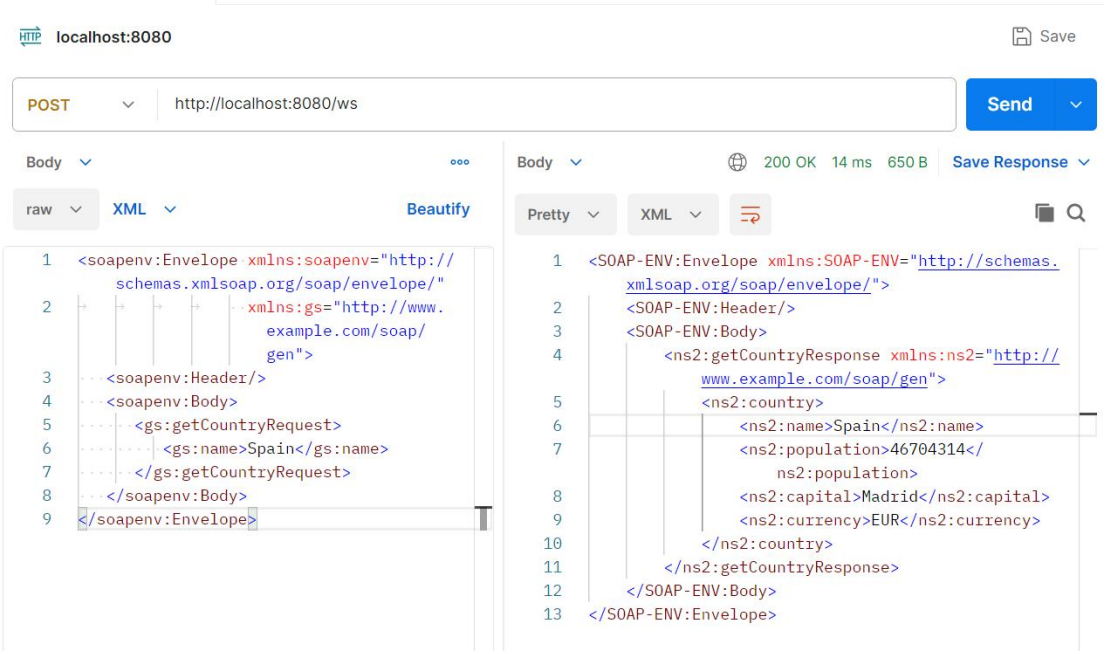
c) Tambahkan SOAP Request berikut ini di tab Body>raw (pilih XML).

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                                    xmlns:gs="http://www.example.com/soap/gen">
  <soapenv:Header/>
  <soapenv:Body>
    <gs:getCountryRequest>
      <gs:name>Spain</gs:name>
    </gs:getCountryRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

d) Klik tombol Send. Setelah itu akan menampilkan response sebagai berikut.

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <ns2:getCountryResponse xmlns:ns2="http://www.example.com/soap/gen">
         <ns2:country>
            <ns2:name>Spain</ns2:name>
            <ns2:population>46704314</ns2:population>
            <ns2:capital>Madrid</ns2:capital>
            <ns2:currency>EUR</ns2:currency>
         </ns2:country>
```

```
    </ns2:getCountryResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```



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.