Panduan Pembuatan Instance XML menggunakan XSD secara Terprogram Menggunakan Bahasa Pemrograman Java

Dokumen ini merupakan bentuk dukungan dari tim pengembang aplikasi DJP bagi pengembang aplikasi pada Lembaga Jasa Keuangan (LJK) maupun konsultan yang menyediakan jasa pengembangan aplikasi kepada LJK. Panduan ini diharapkan dapat membantu mereka dalam mengembangkan aplikasi pembentukan instance document respon permintaan informasi keuangan dalam format XML. Sebelum membaca dokumen ini pastikan anda telah memiliki dan telah membaca dokumentasi XSD respon permintaan informasi keuangan.

Panduan ini akan menjelaskan mengenai bagaimana menulis program dengan bahasa pemrograman Java untuk membuat instance document dan melakukan validasi atas instance document berdasarkan file XSD yang diterbitkan oleh Direktorat Jenderal Pajak (DJP).

A. Membuat *instance document* berdasarkan XSD

Langkah-langkah pembuatan program untuk menulis file XML berdasarkan file XSD yang disediakan oleh DJP adalah sebagai berikut.

1. Membuat Project Java Maven

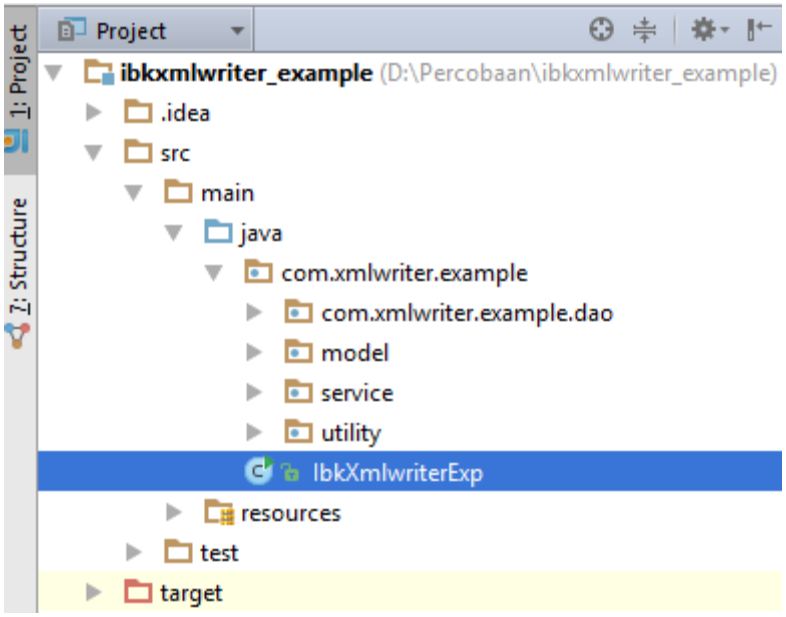
Langkah pertama untuk membuat aplikasi untuk membuat *instance document* XML dan validasi berdasarkan XSD adalah membuat project java maven pada IDE (panduan ini menggunakan Intellij IDE). Beberapa library yang akan digunakan dalam panduan ini harus ditambahkan pada file POM.XML sebagai dependency, yaitu sebagai berikut.

```
<dependency>
    <groupId>com.opencsv</groupId>
    <artifactId>opencsv</artifactId>
    <version>5.3</version>
</dependency>
```

```
<dependency>
    <groupId>org.exist-db.thirdparty.xerces</groupId>
    <artifactId>xercesImpl</artifactId>
    <version>2.12.0</version>
    <classifier>xml-schema-1.1</classifier>
</dependency>
<dependency>
    <groupId>org.exist-db.thirdparty.org.eclipse.wst.xml</groupId>
    <artifactId>xpath2</artifactId>
    <version>1.2.0</version>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>edu.princeton.cup</groupId>
    <artifactId>java-cup</artifactId>
    <version>10k</version>
    <scope>runtime</scope>
</dependency>
```

```
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.11.1</version>
</dependency>
```

Struktur project maven yang dibuat adalah sebagai berikut.



2. Membuat *Plain Old Java Object* (POJO) berdasarkan file XSD

Plain Old Java Object (POJO) merupakan Java Class yang hanya terdiri dari property dan method untuk setter dan getter. Class ini dapat digenerate dari file XSD dengan fitur XJC yang tersedia pada Java Development Kit (JDK). Versi JDK yang digunakan dalam panduan ini adalah 1.8.0_25.

XSD yang diterbitkan DJP mengacu pada XML Schema versi 1.1[1] karena terdapat kebutuhan untuk menerapkan *rule based validation*[2], dan pada versi sebelumnya (1.0) tidak mendukung kemampuan tersebut. Versi XSD tersebut sampai saat ini belum didukung dalam JDK[3]. Oleh karena itu untuk kepentingan *code generation* disediakan file XSD dengan struktur data yang sama namun dalam versi yang tidak menyertakan tag terkait validasi (XSD 1.0).

Code generation (POJO Class) dilakukan dengan menjalankan perintah XJC dengan menyertakan parameter berupa nama package yang dikehendaki kemudian nama file XSD yang digunakan, yaitu sebagai berikut:

[1] https://www.w3.org/TR/xmlschema11-1/
[2] https://blogs.oracle.com/rammenon/xml-schema-11-ndash-what-you-need-to-know
[3] https://bugs.openjdk.java.net/browse/JDK-8197490

```
xjc -p com.xmlwriter.example.model respon-ibk-v1_0_0.xsd
```

Setelah menjalankan perintah tersebut maka akan terbentuk susunan folder sesuai nama package yang kita tentukan dan POJO berupa file berekstensi java terbentuk di dalamnya.

```
parsing a schema...
compiling a schema...
com\xmlwriter\example\model\CurrType.java
com\xmlwriter\example\model\DataKeuanganType.java
com\xmlwriter\example\model\DataRekeningType.java
com\xmlwriter\example\model\InfoRekeningType.java
com\xmlwriter\example\model\LjkType.java
com\xmlwriter\example\model\MutasiRekeningType.java
com\xmlwriter\example\model\ObjectFactory.java
com\xmlwriter\example\model\ResponDataType.java
com\xmlwriter\example\model\ResponPermintaanType.java
com\xmlwriter\example\model\ResponType.java
com\xmlwriter\example\model\SuratJawabanType.java
```

3. Membuat DAO class untuk membaca dari sumber data

Data Access Object merupakan class yang dibuat untuk melakukan operasi ke sumber data. Ruang lingkup aplikasi ini adalah untuk membentuk instance document XML dari XSD yang diterbitkan oleh DJP sehingga kebutuhan pada akses data hanya berupa retrieve/ pencarian dan pengambilan data. DAO class yang dibuat pada panduan ini hanya berupa membaca csv file, pada implementasi sesungguhnya pengembang aplikasi bisa menggantinya dengan pengambilan data ke basis data. Panduan ini mensimulasikan akses data dengan beberapa file CSV berikut:

1) dt_ljk.csv, yaitu data identitas LJK;
2) dt_surat.csv, yaitu data surat jawaban dari LJK
3) dt_wp.csv, yaitu data wp yang dimintakan informasi keuangan oleh DJP dan konfirmasi dari LJK apakah ditemukan sebagai nasabah atau tidak;
4) dt_inforek.csv, yaitu data mengenai rekening keuangan atas wajib pajak yang merupakan nasabah pada LJK;
5) dt_mutasirek.csv, yaitu data mengenai transaksi/ mutasi rekening keuangan atas wajib pajak yang merupakan nasabah pada LJK.

Berdasarkan definisi pada XSD data yang dibutuhkan adalah data identitas LJK, data surat jawaban dan repon atas permintaan informasi keuangan. Sehingga struktur DAO class terdiri dari method - method berikut.

```
public class RekeningDao {
    public RekeningDao(){
```

```
    }

    public List<String[]> getDtLjk(){
        List<String[]> res = new ArrayList<String[]>();
        try{
            CSVReader reader = new CSVReader(new FileReader("./dt_ljk.csv"));
            res = reader.readAll();
        }catch(Exception e){
            System.out.println(e);
        }

        return res;
    }

    public List<String[]> getDtWp(){
        List<String[]> res = new ArrayList<String[]>();
        try{
            CSVReader reader = new CSVReader(new FileReader("./dt_wp.csv"));
            res = reader.readAll();
        }catch(Exception e){
            System.out.println(e);
        }

        return res;
    }

    public List<String[]> getDtSurat(String noSurat){
        List<String[]> res = new ArrayList<String[]>();
        List<String[]> respon = new ArrayList<String[]>();
        try{
            CSVReader reader = new CSVReader(new FileReader("./dt_surat.csv"));
            res = reader.readAll();
            for(String[] surat : res){
                if(surat[2].equals(noSurat)){
                    respon.add(surat);
                }
            }
        }catch(Exception e){
            System.out.println(e);
        }

        return respon;
    }

    public List<String[]> getDtInforekByNik(String nik){
        List<String[]> res = new ArrayList<String[]>();
        try{
            CSVReader reader = new CSVReader(new FileReader("./dt_inforek.csv"));
            List<String[]> lsrek = reader.readAll();
            for(String[] rek : lsrek){
                if(rek[10].equals(nik)){
                    res.add(rek);
                }
            }
        }catch(Exception e){
            System.out.println(e);
        }

        return res;
    }


    public List<String[]> getDtInforekByNpwp(String npwp){
```

```
        List<String[]> res = new ArrayList<String[]>();
        try{
            CSVReader reader = new CSVReader(new FileReader("./dt_inforek.csv"));
            List<String[]> lsrek = reader.readAll();
            for(String[] rek : lsrek){
                if(rek[0].equals(npwp)){
                    res.add(rek);
                }
            }
        }catch(Exception e){
            System.out.println(e);
        }

        return res;
    }

    public List<String[]> getDtMutasirek(String norek){
        List<String[]> res = new ArrayList<String[]>();
        try{
            CSVReader reader = new CSVReader(new FileReader("./dt_mutasirek.csv"));
            List<String[]> lsmutasi = reader.readAll();
            for(String[] mutasi : lsmutasi){
                if(mutasi[0].equals(norek)){
                    res.add(mutasi);
                }
            }
        }catch(Exception e){
            System.out.println(e);
        }

        return res;
    }
}
```

4. Membuat Class untuk mapping data permintaan dari file JSON ke object

Data permintaan informasi keuangan dari DJP dapat diunduh dalam format JSON pada Portal EoI DJP. Data permintaan merupakan kumpulan dari objek data surat permintaan dalam format JSON. FIle tersebut dapat dibaca dengan melakukan mapping ke object dengan library JSON pada Java. Berikut ini adalah class-class yang harus dibuat untuk dapat membaca file tersebut:

a. DataSuratPmt, yaitu class data surat yang berisi header dan detail surat permintaan;
b. MdlPermintaanHdr, yaitu class header surat permintaan;
c. MdlPermintaanDtl, yaitu class detail surat permintaan.

5. Membuat Service class untuk menulis XML

Bagian ini merupakan bagian yang penting dalam panduan ini. Service class ini berperan untuk menulis data yang telah dibaca dari sumber data ke dalam strtuktur data dalam Schema XML (XSD).

a. Membaca data surat permintaan dalam format JSON

```
public List<DataSuratPmt> getDatareq(String pathFile) throws Exception{
    List<DataSuratPmt> respon = new ArrayList<DataSuratPmt>();
    ObjectMapper objectMapper = new ObjectMapper();
    respon = objectMapper.readValue(new File(pathFile),new
TypeReference<List<DataSuratPmt>>(){});
    return respon;
```

```
}

..........

List<DataSuratPmt> dataPmt = getDatareq("data-permintaan-775.json");
```

b. Membuat objek respon

Untuk setiap surat permintaan dibuatkan objek respon. Kemudian setiap objek respon diisi dengan data jawaban.

```
for(DataSuratPmt pmt : dataPmt){
    ResponType respon = new ResponType();

    ..................
    ..................
```

c. Membuat dan mengisi objek LJK

```
//mengambil data ljk
List<String[]> lsLjk = rekDao.getDtLjk();
String[] dtLjk = lsLjk.size()>0 ? lsLjk.get(0): null;

//membentuk objek LJK
LjkType ljk = new LjkType();
ljk.setNpwpLjk(dtLjk[0]);
ljk.setNamaLjk(dtLjk[1]);
```

d. Membuat dan mengisi objek surat jawaban

```
List<String[]> lsSurat = rekDao.getDtSurat(hdrPmt.getNoSurat());
String[] dtSurat = lsSurat.size()>0 ? lsSurat.get(0): null;

//membuat objek surat jawaban
SuratJawabanType suratJawaban = new SuratJawabanType();
suratJawaban.setNoRespon(dtSurat[0]);
XMLGregorianCalendar xmlGregCal = formatter.toXMLGregCal(dtSurat[1]);
suratJawaban.setTglRespon(xmlGregCal);
suratJawaban.setNoSuratPermintaan(hdrPmt.getNoSurat());
suratJawaban.setStatusRespon(dtSurat[3]);
suratJawaban.setNamaPj(dtSurat[4]);
suratJawaban.setJabatanPj(dtSurat[5]);
```

e. Membuat dan mengisi objek respon permintaan dan data keuangan

```
ResponPermintaanType responPermintaan = new ResponPermintaanType();
List<ResponDataType> lsResponData = new ArrayList();
List<MdlPermintaanDtl> lsWp = pmt.getPmtDtls();
System.out.println("4.1. Melakukan pencarian data rekening keuangan untuk
"+lsWp.size()+" WP sesuai permintaan dari DJP");
for(MdlPermintaanDtl wp : lsWp){
    System.out.println("Mencari data rekening keuangan untuk NIK
"+wp.getNikNasabah()+" atau NPWP "+wp.getNpwpNasabah());
    ResponDataType responData = new ResponDataType();
    responData.setNik(wp.getNikNasabah());
    responData.setNpwp(wp.getNpwpNasabah());
    responData.setNamaWp(wp.getNmNasabah());

    //melakukan pencarian rekening keuangan WP
    List<String[]> lsRek = new ArrayList<>();
    if(wp.getNikNasabah() != null || !wp.getNikNasabah().equals("")){
        //apabila pada data WP terdapat NIK
        lsRek = rekDao.getDtInforekByNik(wp.getNikNasabah());
        //apabila tidak ditemukan dengan key NIK maka dicari dengan key NPWP
        if(lsRek.size() == 0){
            lsRek = rekDao.getDtInforekByNpwp(wp.getNpwpNasabah());
        }
    }else{
        lsRek = rekDao.getDtInforekByNpwp(wp.getNpwpNasabah());
    }

    String stsNasabah = lsRek.size() > 0 ? "1" : "0";

    responData.setStatusNasabah(stsNasabah);
```

f. Membuat dan mengisi objek data rekening

```
//apabila WP merupakan nasabah dari LJK
if(stsNasabah.equals("1")){
    DataKeuanganType dataKeuangan = new DataKeuanganType();
    List<DataRekeningType> lsDataRek = new ArrayList<DataRekeningType>();
    for(String[] rek : lsRek){
        DataRekeningType dataRekening = new DataRekeningType();
```

g. Membuat dan mengisi objek info rekening

Setiap rekening yang ditemukan dari hasil pencarian data rekening dengan kunci pencarian NIK atau NPWP dibuatkan objek data rekening yang mencakup informasi rekening dan mutasi rekening

```
for(String[] rek : lsRek){
    DataRekeningType dataRekening = new DataRekeningType();

    InfoRekeningType infoRekening = new InfoRekeningType();
    infoRekening.setCif(rek[1]);
    infoRekening.setMataUang(CurrType.valueOf(rek[2]));
    infoRekening.setStatusRekening(rek[3]);
    infoRekening.setNamaRekening(rek[4]);
    infoRekening.setNoRekening(rek[5]);
    infoRekening.setSaldoAkhir(new BigDecimal(rek[6]));
    infoRekening.setSaldoAwal(new BigDecimal(rek[7]));
```

```
infoRekening.setTglAkhir(formatter.toXMLGregCal(rek[8]));
infoRekening.setTglAwal(formatter.toXMLGregCal(rek[9]));
infoRekening.setTglBukaRek(formatter.toXMLGregCal(rek[10]));
```

h. Membuat dan mengisi objek mutasi rekening

```
List<MutasiRekeningType> lsMutasiRek = new
ArrayList<MutasiRekeningType>();

//mengambil data mutasi dari sumber data dengan parameter nomor rekening
List<String[]> lsMutasi = rekDao.getDtMutasirek(rek[5]);
for(String[] mutasi : lsMutasi){
    MutasiRekeningType mutasiRekening = new MutasiRekeningType();
    mutasiRekening.setNoRekening(mutasi[0]);
    mutasiRekening.setBerita(mutasi[1]);
    mutasiRekening.setKdBankLawan(mutasi[2]);
    mutasiRekening.setKdJnsTrans(mutasi[3]);
    mutasiRekening.setKodeDebitCredit(mutasi[4]);
    mutasiRekening.setNilaiTransaksi(new BigDecimal(mutasi[5]));
    mutasiRekening.setNoRekeningLawan(mutasi[6]);
    mutasiRekening.setTglTransaksi(formatter.toXMLGregCal(mutasi[7]));

    lsMutasiRek.add(mutasiRekening);
}
```

i. Menggabungkan objek info rekening dan mutasi rekening ke dalam objek data rekening

```
dataRekening.setInfoRekening(infoRekening);
dataRekening.getMutasiRekening().addAll(lsMutasiRek);

lsDataRek.add(dataRekening);
```

j. Menggabungkan objek data rekeing ke dalam objek data keuangan

```
dataKeuangan.getDataRekening().addAll(lsDataRek);
```

k. Menggabungkan objek data keuangan ke dalam objek respon data

```
responData.setDataKeuangan(dataKeuangan);
```

l. Menggabungkan seluruh objek respon data ke dalam objek respon permintaan.

```
for(String[] wp : lsWp){
    ..................................................
    ..................................................

    lsResponData.add(responData);

}

responPermintaan.getResponData().addAll(lsResponData);
```

m. Menggabungkan objek ljk, surat jawaban, dan respon permintaan ke dalam objek respon.

```
respon.setLembagaJasaKeuangan(ljk);
respon.setSuratJawaban(suratJawaban);
respon.setResponPermintaan(responPermintaan);
```

n. Marshaling objek respon

Pada tahap ini objek data yang terbentuk ditulis ke dalam file XML pada lokasi dan nama file yang ditentukan.

```
mar.marshal(respon,new File("./responibk"+dtSurat[0]+".xml"));
```

B. Validasi *instance document* menggunakan XSD

Proses pembentukan instance document XML telah dijelaskan pada bagian sebelumnya, kemudian file XML yang telah dibentuk tersebut harus divalidasi untuk memastikan tidak ada kesalahan. Validasi dilakukan berdasarkan file XSD yang lengkap, dan bukan menggunakn file XSD yang less validation digunakan untuk code generation.

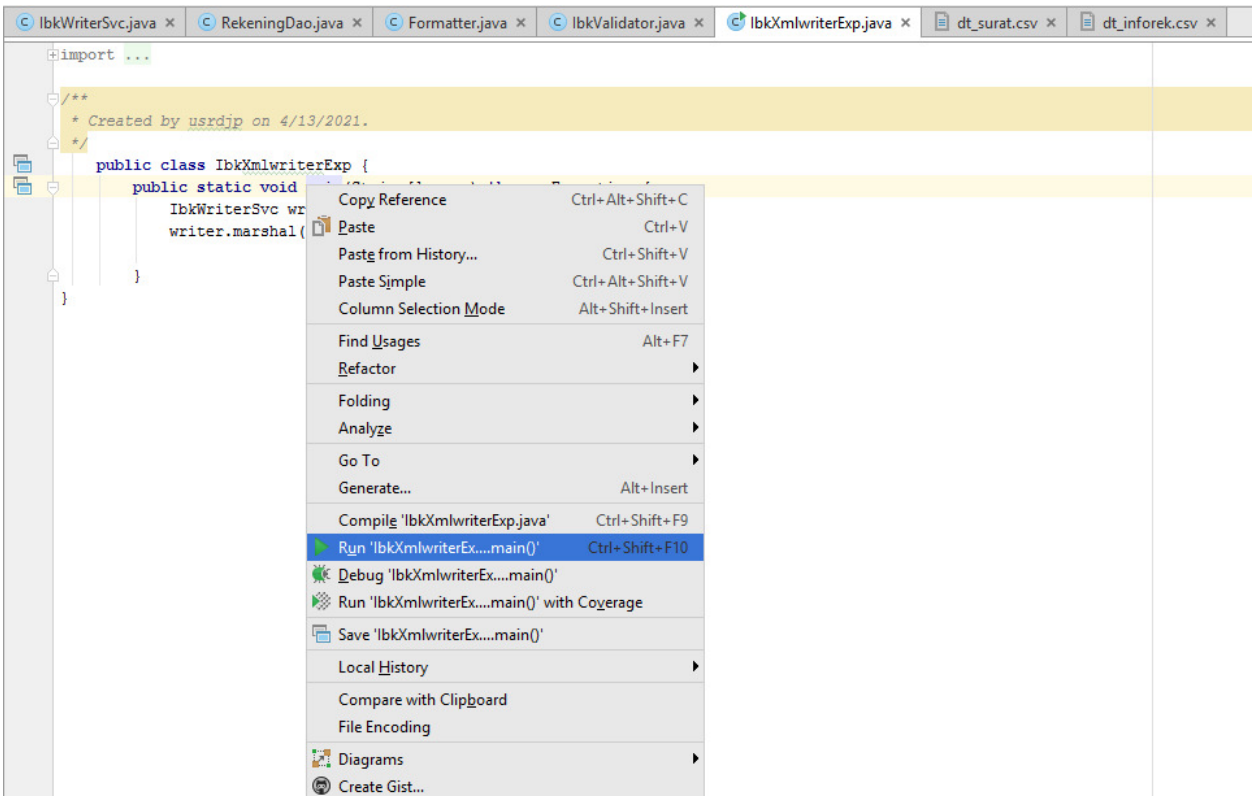
1. Membuat Class Validator dan mendefinisikan fungsi untuk melakukan validasi file XML

```
public void validate(String xsdPath, String xmlPath){
    try {
                SchemaFactory factory =
SchemaFactory.newInstance("http://www.w3.org/XML/XMLSchema/v1.1");
        Schema schema = factory.newSchema(new File(xsdPath));
        Validator validator = schema.newValidator();
        validator.validate(new StreamSource(new File(xmlPath)));
        System.out.println("Instant Document "+xmlPath+" Berhasil dibuat dan telah
tervalidasi");
    } catch (IOException e){
        System.out.println("Exception: "+e.getMessage());
    }catch(SAXParseException e2){
        int line = e2.getLineNumber();
        int col = e2.getColumnNumber();
        String message = e2.getMessage();
        String prsMessage = "Ditemukan Error ketika validasi XML dengan Skema
XSD\n" +
                "baris: " + line + "\n" +
                "kolom: " + col + "\n" +
                "pesan: " + message.substring(message.indexOf(":") + 2);
        System.out.println("Ditemukan Error ketika validasi XML dengant Skema
XSD\n" +
                "baris: " + line + "\n" +
                "kolom: " + col + "\n" +
                "pesan: " + message.substring(message.indexOf(":") + 2));

    }
    catch(SAXException e1){
        System.out.println("SAX Exception: "+e1.getMessage());

    }
}
```

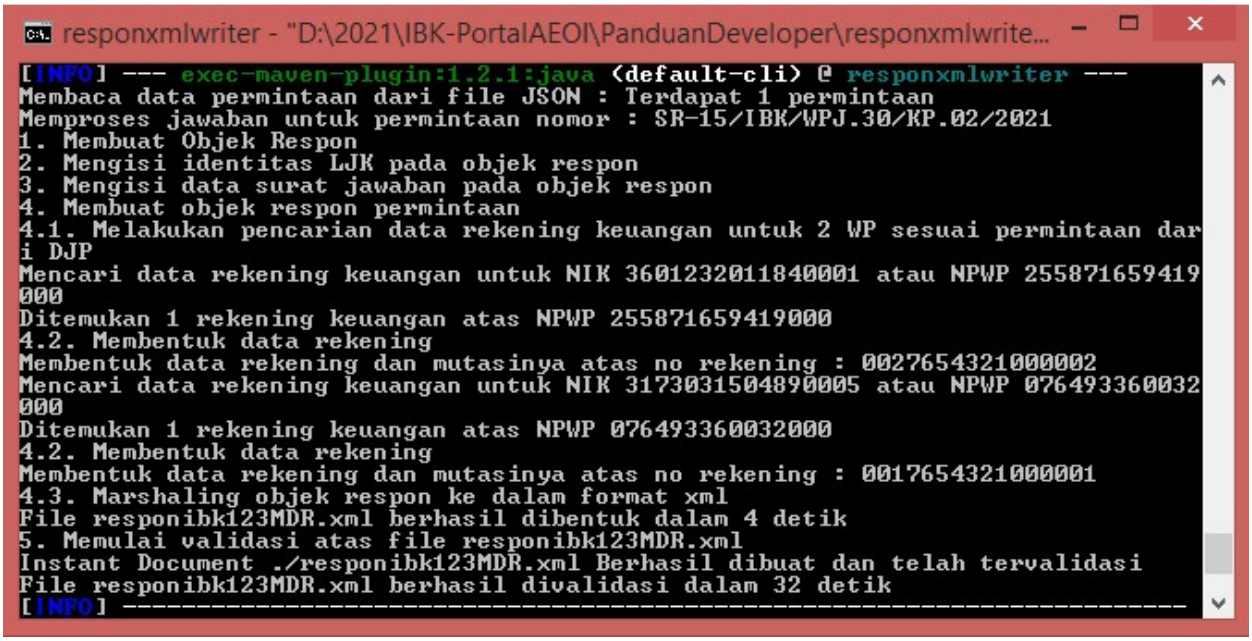
2. Melakukan validasi terhadap file XML yang telah terbentuk

```
ClassLoader classLoader = Thread.currentThread().getContextClassLoader();
URL resource = classLoader.getResource("respon-ibk-v1_1_0.xsd");

IbkValidator validator = new IbkValidator();
validator.validate(resource.getPath(), "./responibk"+dtSurat[0]+".xml");
```

3. Contoh java project untuk menulis file XML berdasarkan XSD.
   Contoh code lengkap dapat ditemukan pada repository developer kami di **https://github.com/digital-budisantoso/responxml-writer**. Setelah code didownload atau diclone lakukan update atau import maven. Selanjutnya buka code dengan IDE. pada panduan ini IDE yang digunakan adalah Intellij. Jalankan code dengan klik kanan pada main class sebagaimana pada gambar di bawah ini.



Aplikasi dapat juga dibuild dan dijalankan dengan perintah maven, yaitu mvn install kemudian mvn exec:java, outputnya sebagaimana pada gambar di bawah ini. Setelah itu, File XML akan terbentuk pada root direktori dari project.



-- Semoga Bermanfaat --