KODE PROGRAM SISTEM INFORMASI PEMBAYARAN PAJAK BUMI DAN BANGUNAN PERDESAAN DAN PERKOTAAN DI KABUPATEN BREBES SERTA PENJELASANNYA

18 April 2018

Priyanto Tamami, S.Kom.

1 PENDAHULUAN

Karena program dibangun secara terpisah antara bagian-belakang (*backend*) dan bagian-depan (*frontend*) serta keduanya menggunakan *framework* yang berbeda, maka kode program dan penjelasannya akan terbagi menjadi 2 (dua) bagian yang nantinya akan dibahas pada seksi masing-masing.

2 BAGIAN-BELAKANG (*BACKEND*)

Bagian-belakang (*backend*) dibangun menggunakan Springboot dimana didalamnya sudah terdapat *servlet container* yang telah ditanamkan dan dapat dijalankan langsung dalam paket berkas berformat `jar` tanpa harus melakukan *deploy* pada *servlet container* yang lain. Berikut adalah kelas-kelas yang membangun sistem aplikasi ini sehingga dapat beroperasi sebagaimana mestinya.

2.1 Kelas `ApiController`

Kelas ini merupakan gerbang depan dari sistem *backend*, yang berisi layanan-layanan *service* berupa REST API (*Representational State Transfer - Application*

Programmable Interface), hal ini ditandai dengan adanya kode berikut di atas deklarasi kelas :

```
1 @RestController
```

Langkah pertama adalah menyiapkan *log* untuk kelas ini sehingga mempermudah diagnosa apabila terjadi kesalahan dalam operasional aplikasi ini nantinya. Berikut adalah deklarasi pembentukan *log* pada kelas ini :

```
1 private static Logger logger = Logger.getLogger(ApiController.class);
```

Kelas ini pun akan menggunakan beberapa *service* untuk keperluan melakukan pemilahan data-data mana saja yang nantinya akan disampaikan ke bagian-depan (*frontend*) untuk ditampilkan. Deklarasi penggunaan *service* ini seperti ditunjukkan pada kode berikut :

```
1 @Autowired
2 private DatObjekPajakService datOpService;
3
4 @Autowired
5 private DatSubjekPajakService datSpService;
6
7 @Autowired
8 private SpptService spptService;
```

2.1.1 *Method* `getDataOp()`

API (*Application Programmeable Interface*) yang pertama dari kelas ini adalah `android_service` seperti ditunjukkan dengan deklarasi berikut :

```
1 @RequestMapping(value = "/android_service", method = RequestMethod.
    POST)
2 public ReturnData getDataOp2(@RequestBody String data,
    HttpServletResponse response) {}
```

Pada baris pertama dari kode tersebut adalah menyiapkan API (*Application Programmable Interface*) dari *android_service* dengan *method* permintaan datanya berupa `POST`.

Pada baris kedua, *method* ini akan mengembalikan kelas `ReturnData` dalam bentuk JSON yang terkonversi secara otomatis, sedangkan data yang diterima dari klien akan masuk dalam variabel `data` dalam tipe `String`.

Variabel `response` dengan tipe `HttpServletResponse` digunakan untuk melakukan pengiriman status akses bahwa seluruh klien dapat melakukan koneksi ke peladen API ini dari manapun.

Selanjutnya menyiapkan beberapa variabel yang akan digunakan untuk memilah bentuk permintaan dari klien dan mempersiapkan data yang akan digunakan sebagai parameter pengambilan data pada sistem basis data. Berikut ada penggalan kode persiapan variabel yang akan digunakan :

```
1 String keyword = "", nop = "", subjekPajakId="", tahun="";
2 String [] pairs = data.split("\\&");
3 HashMap<String, String> param = new HashMap<>();
```

Variabel `keyword` nantinya digunakan untuk membedakan jenis permintaan data yang diinginkan, variabel `nop` digunakan untuk menyimpan parameter Nomor Objek Pajak (NOP), variabel `subjekPajakId` digunakan untuk menyimpan identitas wajib pajak, dan variabel `tahun` digunakan untuk menyimpan parameter permintaan data untuk tahun pajak.

Variabel `pairs` digunakan sebagai varibel sementara untuk menyimpan pasangan data *key* dan *value* dari parameter *request* yang dikirimkan oleh klien dalam format `key=value` dalam sebuah larik.

Variabel `param` memiliki tipe data `HashMap` karena fungsinya akan menyimpan pasangan data pada larik `pairs` menjadi bentuk struktur data naturalnya yang terdiri `key` dan `value`.

Potongan kode berikutnya adalah seperti ini :

```
1 for(int i=0; i<pairs.length; i++) {
2   param.put(pairs[i].split("=")[0], pairs[i].split("=")[1]);
3 }
```

Potongan kode tersebut sebetulnya hanya memindahkan isi data larik `pairs` kedalam tipe data `Map` yang memang ditujukan untuk pembentukan rangkaian pasangan data `key` dan `value`.

Blok kode berikutnya adalah mengambil nilai dari masing-masing parameter *request* yang dikirimkan oleh klien untuk digunakan sebagai parameter *request* ke sistem basis data. Berikut kode programnya :

```
1 keyword = param.get("keyword"); param.remove("keyword");
2 nop = param.get("nop"); param.remove("nop");
3 subjekPajakId = param.get("subjek_pajak_id"); param.remove("
    subjek_pajak_id");
4 tahun = param.get("tahun"); param.remove("tahun");
```

Pada baris pertama dari kode tersebut, variabel `keyword` akan mengambil nilai atau isi atau `value` dari variabel `param` dengan kunci atau `key` berupa teks `keyword`.

Untuk variabel `nop`, `subjekPajakId`, dan `tahun`, masing-masing pun akan terisi dengan nilai dari variabel `param` dengan kunci masing-masing berupa `nop`, `subjek_pajak_id` dan `tahun`.

Kode berikutnya sangat sederhana, terdiri hanya 1 (satu) baris saja, fungsinya untuk memberikan *header* pada *response body* agar klien dapat melakukan akses terhadap API ini. Kodenya adalah seperti berikut :

```
1 response.setHeader("Access-Control-Allow-Origin", "*");
```

Langkah berikutnya adalah melakukan seleksi terhadap isi variabel `keyword`, berikut adalah cuplikan kodenya :

```
1 if(keyword.equals("op")) {
2   logger.trace("Request OP untuk NOP : " + nop);
3   return datOpService.getOp(nop);
4 } else if(keyword.equals("wp")) {
5   logger.trace("Request SP untuk ID : " + subjekPajakId);
6   return datSpService.getSubjekPajak(subjekPajakId);
7 } else if(keyword.equals("bayar")) {
8   logger.trace("Data Piutang untuk NOP : " + nop + " / " + tahun);
9   return spptService.getPiutang(nop, tahun);
10 }
```

Pada baris pertama adalah memeriksa apakah isi variabel `keyword` sama dengan `op` atau tidak, bila sama, maka akan melakukan perintah pada baris ke-2 yang melakukan pencatatan ke *log* dan ke-3 yang akan melakukan pengambilan data ke kelas `DatOpService`.

Bila variabel `keyword` berisi teks `wp`, maka akan menjalankan proses pada baris ke-5, yang juga melakukan pencatatan ke dalam *log*, kemudian pada baris ke-6 yang akan memanggil *method* `getSubjekPajak()` milik kelas `DatSpService`.

Pemeriksaan terakhir adalah apabila nilai dari variabel `keyword` berisi teks `bayar`. Langkahnya akan sama dimana pada baris ke-8 sistem akan mencatatkan aktivitasnya pada *log* dan pada baris ke-9 akan memanggil *method* `getPiutang()` milik kelas `SpptService`.

2.2 Kelas `DatObjekPajak`

Kelas ini digunakan untuk `mapping` data dari sistem basis data, sehingga isinya hanya berupa deklarasi properti yang mirip dengan struktur tabel pada sistem basis data, berikut adalah isinya :

```
1 @Id @Getter @Setter
2 private String kdPropinsi;
```

```
3 @Id @Getter @Setter
4 private String kdDati2;
5 @Id @Getter @Setter
6 private String kdKecamatan;
7 @Id @Getter @Setter
8 private String kdKelurahan;
9 @Id @Getter @Setter
10 private String kdBlok;
11 @Id @Getter @Setter
12 private String noUrut;
13 @Id @Getter @Setter
14 private String kdJnsOp;
15 @Getter @Setter
16 private String subjekPajakId;
17 @Getter @Setter
18 private String noFormulirSpop;
19 @Getter @Setter
20 private String noPersil;
21 @Getter @Setter
22 private String jalanOp;
23 @Getter @Setter
24 private String blokKavNoOp;
25 @Getter @Setter
26 private String rwOp;
27 @Getter @Setter
28 private String rtOp;
29 @Getter @Setter
30 private int kdStatusCabang;
31 @Getter @Setter
32 private char kdStatusWp;
33 @Getter @Setter
34 private BigDecimal totalLuasBumi;
35 @Getter @Setter
```

```
36 private BigDecimal totalLuasBng;
37 @Getter @Setter
38 private BigDecimal njopBumi;
39 @Getter @Setter
40 private BigDecimal njopBng;
41 @Getter @Setter
42 private int statusPetaOp;
43 @Getter @Setter
44 private char jnsTransaksiOp;
45 @Getter @Setter
46 private Date tglPendataanOp;
47 @Getter @Setter
48 private String nipPendata;
49 @Getter @Setter
50 private Date tglPemeriksaanOp;
51 @Getter @Setter
52 private String nipPemeriksaOp;
53 @Getter @Setter
54 private Date tglPerekamanOp;
55 @Getter @Setter
56 private String nipPerekamOp;
57
58 @ManyToOne
59 @JoinColumns({
60   @JoinColumn(name = "kdPropinsi", referencedColumnName = "kdPropinsi
      ", insertable = false, updatable = false),
61   @JoinColumn(name = "kdDati2", referencedColumnName = "kdDati2",
      insertable = false, updatable = false),
62   @JoinColumn(name = "kdKecamatan", referencedColumnName = "
      kdKecamatan", insertable = false, updatable = false)
63 })
64 @Getter @Setter
65 private RefKecamatan refKecamatan;
```

```
66
67 @ManyToOne
68 @JoinColumns({
69   @JoinColumn(name="kdPropinsi", referencedColumnName = "kdPropinsi",
       insertable = false, updatable = false),
70   @JoinColumn(name="kdDati2", referencedColumnName = "kdDati2",
      insertable = false, updatable = false),
71   @JoinColumn(name="kdKecamatan", referencedColumnName = "kdKecamatan
      ", insertable = false, updatable = false),
72   @JoinColumn(name = "kdKelurahan", referencedColumnName = "
      kdKelurahan", insertable = false, updatable = false)
73 })
74 @Getter @Setter
75 private RefKelurahan refKelurahan;
```

2.3 Kelas `DatObjekPajakPK`

Kelas ini dibentuk untuk menjadi tempat *mapping* pada *primary key* milik kelas `DatObjekPajak`. Kelas ini dibutuhkan karena tabel `dat_objek_pajak` memiliki banyak *field* untuk membentuk satu *primary key*, sehingga dalam sistem Spring Data JPA, perlu untuk mendeklarasikan kelas tersendiri untuk *mapping* tiap *field* yang menjadi *primary key*, dengan istilah lain pada Spring Data JPA ini adalah *composite id*.

Deklarasi kelas ini pun cukup sederhana, hanya menyiapkan beberapa variabel yang akan memetakan dan menjadi pasangan *field* pada tabel di sistem basis data. Berikut adalah deklarasi dari tiap properti pada kelas `DatObjekPajakPK` :

```
1 @Getter @Setter
2 private String kdPropinsi;
3 @Getter @Setter
4 private String kdDati2;
5 @Getter @Setter
```

```
6  private String kdKecamatan;
7  @Getter @Setter
8  private String kdKelurahan;
9  @Getter @Setter
10 private String kdBlok;
11 @Getter @Setter
12 private String noUrut;
13 @Getter @Setter
14 private String kdJnsOp;
```

2.4 Kelas `DatSubjekPajak`

Kelas ini berfungsi untuk *mapping* data dengan tabel `dat_subjek_pajak` dengan isi properti mirip dengan *field* milik tabel tersebut. Berikut adalah daftar properti yang dimiliki oleh kelas `DatSubjekPajak` :

```
1     @Id @Getter @Setter
2     private String subjekPajakId;
3     @Getter @Setter
4     private String nmWp;
5     @Getter @Setter
6     private String jalanWp;
7     @Getter @Setter
8     private String blokKavNoWp;
9     @Getter @Setter
10    private String rwWp;
11    @Getter @Setter
12    private String rtWp;
13    @Getter @Setter
14    private String kelurahanWp;
15    @Getter @Setter
16    private String kotaWp;
17    @Getter @Setter
```

```
18      private String kdPosWp;
19      @Getter @Setter
20      private String telpWp;
21      @Getter @Setter
22      private String npwp;
23      @Getter @Setter
24      private char statusPekerjaanWp;
```

2.5 Kelas `JsonObject`

Kelas ini nantinya akan memetakan parameter *request* dari klien yang dikirimkan dalam format JSON, isi propertinya adalah seperti berikut ini :

```
1       @Getter @Setter
2       private String keyword;
3       @Getter @Setter
4       private String nop;
5       @Getter @Setter
6       private String subjekPajakId;
7       @Getter @Setter
8       private String tahun;
```

2.6 Kelas `RefKecamatan`

Kelas ini digunakan untuk *mapping* tabel `ref_kecamatan` seingga isi propertinya mirip dengan *field* milik tabel `ref_kecamatan` pada sistem basis data, berikut adalah daftar propertinya (isi kodenya) :

```
1       @Id @Getter @Setter
2       private String kdPropinsi;
3       @Id @Getter @Setter
4       private String kdDati2;
5       @Id @Getter @Setter
```

```
6     private String kdKecamatan;
7     @Getter @Setter
8     private String nmKecamatan;
```

2.7 Kelas `RefKecamatanPK`

Kelas ini akan menjadi definisi bagi *primary key* milik tabel `ref_kecamatan` yang akan digunakan oleh kelas `RefKecamatan`, berikut adalah isi kodenya :

```
1     private String kdPropinsi;
2     private String kdDati2;
3     private String kdKecamatan;
```

2.8 Kelas `RefKelurahan`

Kelas ini merupakan kelas *mapping* untuk tabel `ref_kelurahan` sehingga isi propertinya mirip dengan milik tabel `ref_kelurahan`, berikut adalah isi properti dari kelas `RefKelurahan` :

```
1     @Id @Getter @Setter
2     private String kdPropinsi;
3     @Id @Getter @Setter @Column(insertable = false, updatable = false
      )
4     private String kdDati2;
5     @Id @Getter @Setter
6     private String kdKecamatan;
7     @Id @Getter @Setter
8     private String kdKelurahan;
9     @Getter @Setter
10    private String kdSektor;
11    @Getter @Setter
12    private String nmKelurahan;
13    @Getter @Setter
```

```
14      private BigDecimal noKelurahan;
15      @Getter @Setter
16      private String kdPosKelurahan;
```

2.9 Kelas `RefKelurahanPK`

Kelas ini ditujukan untuk `mapping` definisi *primary key* milik tabel `ref_kelurahan`, berikut adalah isi propertinya dari kelas `RefKelurahanPK` :

```
1       @Getter @Setter
2       private String kdPropinsi;
3       @Getter @Setter
4       private String kdDati2;
5       @Getter @Setter
6       private String kdKecamatan;
7       @Getter @Setter
8       private String kdKelurahan;
```

2.10 Kelas `Sppt`

Kelas ini digunakan untuk *mapping* tabel `sppt` pada sistem basis data, sehingga isinya mirip dengan *field* pada tabel tersebut. Isi properti dari kelas `Sppt` ini adalah seperti berikut :

```
1       @Id @Getter @Setter
2       private String kdPropinsi;
3       @Id @Getter @Setter
4       private String kdDati2;
5       @Id @Getter @Setter
6       private String kdKecamatan;
7       @Id @Getter @Setter
8       private String kdKelurahan;
9       @Id @Getter @Setter
```

```
10      private String kdBlok;
11      @Id @Getter @Setter
12      private String noUrut;
13      @Id @Getter @Setter
14      private String kdJnsOp;
15      @Id @Getter @Setter
16      private String thnPajakSppt;
17      @Getter @Setter
18      private int siklusSppt;
19      @Getter @Setter
20      private String kdKanwilBank;
21      @Getter @Setter
22      private String kdKppbbBank;
23      @Getter @Setter
24      private String kdBankTunggal;
25      @Getter @Setter
26      private String kdBankPersepsi;
27      @Getter @Setter
28      private String kdTp;
29      @Getter @Setter
30      private String nmWpSppt;
31      @Getter @Setter
32      private String jlnWpSppt;
33      @Getter @Setter
34      private String blokKavNoWpSppt;
35      @Getter @Setter
36      private String rwWpSppt;
37      @Getter @Setter
38      private String rtWpSppt;
39      @Getter @Setter
40      private String kelurahanWpSppt;
41      @Getter @Setter
42      private String kotaWpSppt;
```

```
43      @Getter @Setter
44      private String kdPosWpSppt;
45      @Getter @Setter
46      private String npwpSppt;
47      @Getter @Setter
48      private String noPersilSppt;
49      @Getter @Setter
50      private String kdKlsTanah;
51      @Getter @Setter
52      private String thnAwalKlsTanah;
53      @Getter @Setter
54      private String kdKlsBng;
55      @Getter @Setter
56      private String thnAwalKlsBng;
57      @Getter @Setter
58      private Date tglJatuhTempoSppt;
59      @Getter @Setter
60      private BigDecimal luasBumiSppt;
61      @Getter @Setter
62      private BigDecimal luasBngSppt;
63      @Getter @Setter
64      private BigDecimal njopBumiSppt;
65      @Getter @Setter
66      private BigDecimal njopBngSppt;
67      @Getter @Setter
68      private BigDecimal njopSppt;
69      @Getter @Setter
70      private BigDecimal njoptkpSppt;
71      @Getter @Setter
72      private BigDecimal njkpSppt;
73      @Getter @Setter
74      private BigDecimal pbbTerhutangSppt;
75      @Getter @Setter
```

```
76      private BigDecimal faktorPengurangSppt;
77      @Getter @Setter
78      private BigDecimal pbbYgHarusDibayarSppt;
79      @Getter @Setter
80      private char statusPembayaranSppt;
81      @Getter @Setter
82      private char statusTagihanSppt;
83      @Getter @Setter
84      private char statusCetakSppt;
85      @Getter @Setter
86      private Date tglTerbitSppt;
87      @Getter @Setter
88      private Date tglCetakSppt;
89      @Getter @Setter
90      private String nipPencetakSppt;
```

2.11 Kelas `SpptPK`

Digunakan untuk mendefinisikan *primary key* dari tabel `sppt` milik kelas `Sppt`, karena *primary key* milik tabel `sppt` ini terdiri dari beberapa *field*. Berikut adalah isi properti dari kelas `SpptPK` ini :

```
1       @Getter @Setter
2       private String kdPropinsi;
3       @Getter @Setter
4       private String kdDati2;
5       @Getter @Setter
6       private String kdKecamatan;
7       @Getter @Setter
8       private String kdKelurahan;
9       @Getter @Setter
10      private String kdBlok;
11      @Getter @Setter
```

```
12      private String noUrut;
13      @Getter @Setter
14      private String kdJnsOp;
15      @Getter @Setter
16      private String thnPajakSppt;
```

2.12 *Interface* `DatObjekPajakRepo`

Interface ini digunakan untuk melakukan operasi terhadap data pada tabel `dat_objek_pajak`, baik itu tambah data, ubah data, dan hapus data. Karena ini adalah *interface*, maka hanya berisi deklarasi *method* saja.

Bentuk deklarasi *method* pada *interface* ini mengikuti aturan yang diberikan oleh Spring Data JPA untuk melakukan berbagai macam operasi terhadap data pada tabel di sistem basis data. Hanya ada 1 (satu) *method* yang ada pada *interface* ini, berikut adalah kodenya :

```
1     DatObjekPajak findByKdKecamatanAndKdKelurahanAndKdBlokAndNoUrut(
      String kdKecamatan, String kdKelurahan, String kdBlok, String
      noUrut);
```

Kode tersebut untuk mencari data pada tabel `dat_objek_pajak` dengan filter berupa *field* `kd_kecamatan`, `kd_kelurahan`, `kd_blok`, dan `no_urut`

2.13 *Interface* `DatSubjekPajakRepo`

Interface ini digunakan untuk melakukan operasi pada tabel `dat_subjek_pajak`, baik itu tambah data, ubah data, ataupun hapus data. *Interface* ini pun mengikuti aturan dari Spring Data JPA untuk melakukan manipulasi data pada tabel `dat_subjek_pajak`.

Karena ini adalah *interface*, maka hanya berisi deklarasi *method* saja. Berikut adalah daftar *method* yang ada pada *interface* ini :

```
1     DatSubjekPajak findBySubjekPajakId(String subjekPajakId);
2     List<DatSubjekPajak> findBySubjekPajakIdLike(String subjekPajakId
      );
3     List<DatSubjekPajak> findByNmWpLike(String nmWp);
```

Method yang pertama digunakan untuk mencari subjek pajak berdasarkan nomor identitasnya. *Method* kedua akan menghasilkan daftar subjek pajak dengan melakukan filter pencarian berdasarkan potongan nomor identitas subjek pajak. *Method* yang ketiga akan menghasilkan daftar subjek pajak yang pencariannya menggunakan filter nama subjek pajak.

2.14 *Interface* `SpptRepo`

Interface ini pun mengikuti aturan dari Spring Data JPA, dimana didalamnya hanya berisi deklarasi-deklarasi *method* yang fungsinya adalah melakukan manipulasi data pada tabel `sppt` pada sistem basis data, baik berupa tambah data, ubah data, atau hapus data. Deklarasi kode yang terdapat pada *interface* ini adalah seperti berikut :

```
1     List<Sppt> findByKdPropinsiAndKdDati2AndKdKecamatanAnd
2 KdKelurahanAndKdBlokAndNoUrutAndKdJnsOp(
3           String kdPropinsi, String kdDati2, String kdKecamatan,
4           String kdKelurahan, String kdBlok, String noUrut,
5           String kdJnsOp, Sort sort
6     );
7
8     List<Sppt> findByKdPropinsiAndKdDati2AndKdKecamatanAnd
9 KdKelurahanAndKdBlokAndNoUrutAndKdJnsOpAndStatusPembayaranSpptIn(
10          String kdPropinsi, String kdDati2, String kdKecamatan,
11          String kdKelurahan, String kdBlok, String noUrut,
12          String kdJnsOp, Collection<Character> status
13    );
```

```
14
15      Sppt findOneByKdPropinsiAndKdDati2AndKdKecamatanAnd
16 KdKelurahanAndKdBlokAndNoUrutAndKdJnsOpAndThnPajakSpptAnd
17 StatusPembayaranSppt(
18          String kdPropinsi, String kdDati2, String kdKecamatan,
19          String kdKelurahan, String kdBlok, String noUrut,
20          String kdJnsOp, String thnPajak, char statusPembayaranSppt
21      );
```

Method yang pertama digunakan untuk melakukan pengambilan daftar Surat Pemberitahuan Pajak Terhutang (SPPT) dengan filter berdasarkan *field* `kd_propinsi`, `kd_dati2`, `kd_kecamatan`, `kd_kelurahan`, `kd_blok`, `no_urut`, dan `kd_jns_op`, kemudian hasilnya akan dilakukan pengurutan berdasarkan *field* tertentu.

Method yang kedua digunakan untuk melakukan pengambilan daftar Surat Pemberitahuan Pajak Terhutang (SPPT) dengan berdasarkan beberapa *field* yaitu *field* `kd_propinsi`, `kd_dati2`, `kd_kecamatan`, `kd_kelurahan`, `kd_blok`, `no_urut`, `kd_jns_op`, dan `status_pembayaran_sppt`.

Method yang ketiga adalah mengambil data sebuah Surat Pemberitahuan Pajak Terhutang (SPPT) dengan filter berupa *field* `kd_propinsi`, `kd_dati2`, `kd_kecamatan`, `kd_kelurahan`, `kd_blok`, `no_urut`, `kd_jns_op`, `thn_pajak_sppt`, dan `status_pembayaran_sppt`.

2.15 Kelas `DatObjekPajakService`

Kelas ini bertujuan untuk menjembatani komunikasi antara *interface* atau API (*Application Programmable Interface*) sebagai tempat awal masuknya komunikasi bagian-depan (*frontend*), dengan *repository* yang bertugas melakukan manipulasi data pada sistem basis data.

Adakalanya data yang diterima dari bagian-depan (*frontend*) perlu diolah lebih dahulu sebelum akhirnya dapat melakukan manipulasi data pada tingkat / lapisan sistem basis data, atau sebaliknya, dimana kondisi data dari sistem basis

data, perlu dioleh lebih dahulu mana yang akan diteruskan ke bagian-depan (*frontend*) mana yang tidak perlu, dalam hal ini tentunya melihat efisiensi lebar-pita (*bandwidth*) yang nantinya digunakan.

Baris awal adalah mendeklarasikan *repository* yang akan digunakan, berikut kodenya :

```
1     @Autowired
2     private DatObjekPajakRepo datOpRepo;
```

Berikutnya hanya ada sebuah *method* yang fungsinya akan mengembalikan data dari tabel `dat_objek_pajak` namun telah dimodifikasi sesuai kebutuhan bagian-depan *frontend*, berikut adalah kodenya :

```
1     public DatObjekPajakModif getOp(String nop) {
2         DatObjekPajakPK pk = new DatObjekPajakPK();
3         pk.setKdPropinsi(nop.substring(0,2));
4         pk.setKdDati2(nop.substring(2,4));
5         pk.setKdKecamatan(nop.substring(4,7));
6         pk.setKdKelurahan(nop.substring(7,10));
7         pk.setKdBlok(nop.substring(10,13));
8         pk.setNoUrut(nop.substring(13,17));
9         pk.setKdJnsOp(nop.substring(17,18));
10
11        DatObjekPajak data = datOpRepo.findOne(pk);
12
13        return new DatObjekPajakModif(
14                data.getRefKecamatan().getNmKecamatan(),
15                data.getRefKelurahan().getNmKelurahan(),
16                data.getJalanOp(), "RT. " + data.getRtOp() +
17                " RW. " + data.getRwOp(), data.getTotalLuasBumi(),
18                data.getTotalLuasBng(), data.getNjopBumi(),
19                data.getNjopBng(), data.getSubjekPajakId());
20    }
```

Method ini membutuhkan sebuah parameter Nomor Objek Pajak (NOP) agar dapat mengembalikan sebuah nilai.

2.16 Kelas `DatSubjekPajakService`

Kelas ini pun menjadi penghubung antara *interface* luar yang menghubungkan bagian-depan (*frontend*) dan bagian-belakang (*backend*) sebagai kelas yang menyediakan adaptasi data untuk operasi data pada tabel `dat_subjek_pajak`.

Pada awal-awal baris kode, berisi deklarasi kode untuk membuat instan dari *interface* `DatSubjekPajakRepo`, berikut adalah baris kode tersebut :

```
1     @Autowired
2     private DatSubjekPajakRepo datSubjekPajakRepo;
```

Kode berikutnya adalah sebuah *method* yang akan memberikan data dari tabel `dat_subjek_pajak` setelah melakukan penyesuaian data bergantung kebutuhan dari bagian-depan (*frontend*). Berikut ada kode nya :

```
1     public DatSubjekPajakModif getSubjekPajak(String subjekPajakId) {
2         DatSubjekPajak data = datSubjekPajakRepo.findBySubjekPajakId(
3             String.format("%-30s", subjekPajakId));
4
5         return new DatSubjekPajakModif(data.getNmWp(),
6             data.getJalanWp(), "RT. " + data.getRtWp() + " RW. " +
7             data.getRwWp(), data.getKelurahanWp(), data.getKotaWp());
8     }
```

Method ini membutuhkan sebuah parameter bertipe `String` yang berisi nomor identitas subjek pajak.

2.17 Kelas `SpptService`

Kelas ini seperti kelas *service* lainnya, yaitu menghubungkan antara *interface* tempat bagian-depan (*frontend*) berkomunikasi dengan bagian *repository* tempat manipulasi sistem basis data terjadi.

Kode awal dari kelas ini adalah mendeklarasikan dan membentuk instan dari *interface* `SpptRepo` seperti berikut :

```
1     @Autowired
2     private SpptRepo spptRepo;
```

Bagian kedua dari blok kode program pada kelas ini adalah deklarasi *method* `getSppt()` yang berfungsi untuk mengambil daftar Surat Pemberitahuan Pajak Terhutang (SPPT) untuk sebuah Nomor Objek Pajak (NOP) pada tiap tahun pajaknya, maka dari itu *method* ini membutuhkan sebuah parameter bertipe teks (`String`) yang berisi Nomor Objek Pajak (NOP) tanpa tanda baca lain. Berikut adalah deklarasinya :

```
1     public List<SpptModif> getSppt(String nop) {
2         List<SpptModif> result = new LinkedList();
3
4         List<Sppt> data = spptRepo.findByKdPropinsiAndKdDati2
5 AndKdKecamatanAndKdKelurahanAndKdBlokAndNoUrutAndKdJnsOp
6 AndStatusPembayaranSpptIn(
7             nop.substring(0,2), nop.substring(2,4),
8             nop.substring(4,7), nop.substring(7,10),
9             nop.substring(10,13), nop.substring(13,17),
10            nop.substring(17,18),
11            new ArrayList<Character>() {{add('0'); add('1');}}
12        );
13
14        for(int i=0; i<data.size(); i++) {
15            result.add(new SpptModif(data.get(i).getThnPajakSppt(),
```

```
                data.get(i).getPbbYgHarusDibayarSppt(),
                data.get(i).getStatusPembayaranSppt() == '1' ?
                    "LUNAS" : "BLM BAYAR"));
        }

        return result;
    }
```

Bagian ketiga adalah blok kode program tentang *method* yang akan mengembalikan nilai piutang dari sebuah Surat Pemberitahuan Pajak Terhutang (SPPT) apabila status dari Surat Pemberitahuan Pajak Terhutang (SPPT) tersebut memang belum dibayarkan. Berikut adalah kode programnya :

```
    public PiutangSpptModif getPiutang(String nop, String thnPajak) {
        DateTimeFormatter dateFormatOut =
            DateTimeFormat.forPattern("dd-MM-yyyy");

        Sppt data = spptRepo.findOneByKdPropinsiAndKdDati2
AndKdKecamatanAndKdKelurahanAndKdBlokAndNoUrutAndKdJnsOp
AndThnPajakSpptAndStatusPembayaranSppt(
            nop.substring(0,2), nop.substring(2,4),
            nop.substring(4,7), nop.substring(7,10),
            nop.substring(10,13), nop.substring(17,18),
            thnPajak, '0'
        );

        PiutangSpptModif result = new PiutangSpptModif();
        DateTime tglJatuhTempo = new DateTime(
            data.getTglJatuhTempoSppt());
        result.setTgl_tempo(dateFormatOut.print(tglJatuhTempo));
        result.setJumlah(data.getPbbYgHarusDibayarSppt());
        result.setTgl_skrg(dateFormatOut.print(new DateTime()));
        int selisih = Months.monthsBetween(
```

```
21              new DateTime(data.getTglJatuhTempoSppt()),
22              new DateTime()).getMonths();
23          result.setDenda_bln(selisih);
24          BigDecimal denda;
25
26          if(selisih >= 0)
27              result.setDenda_jml(
28                  (selisih > 15) ?
29                      (new BigDecimal("15").multiply(
30                          new BigDecimal("0.02"))
31                              .multiply(
32                                  data.getPbbYgHarusDibayarSppt()))
33                              .round(new MathContext(3,
34                                  RoundingMode.UP)) :
35                      (new BigDecimal(selisih)
36                          .multiply(new BigDecimal("0.02"))
37                          .multiply(data.getPbbYgHarusDibayarSppt()))
38                          .round(new MathContext(3, RoundingMode.UP))
39              );
40          else result.setDenda_jml(new BigDecimal("0"));
41
42          return result;
43      }
```

2.18 Kelas `DatObjekPajakModif`

Kelas ini digunakan untuk mengemas data respon dari tabel `dat_objek_pajak` agar sesuai dengan kebutuhan bagian-depan (*frontend*), sehingga isi dari kelas ini hanyalah beberapa properti yang dibutuhkan oleh bagian-depan (*frontend*), berikut ada kodenya :

```
1       @Getter @Setter @NonNull
2       private String op_kec;
```

```
3     @Getter @Setter @NonNull
4     private String op_kel;
5     @Getter @Setter @NonNull
6     private String op_jalan;
7     @Getter @Setter @NonNull
8     private String op_rtrw;
9     @Getter @Setter @NonNull
10    private BigDecimal op_luas_bumi;
11    @Getter @Setter @NonNull
12    private BigDecimal op_luas_bng;
13    @Getter @Setter @NonNull
14    private BigDecimal op_njop_bumi;
15    @Getter @Setter @NonNull
16    private BigDecimal op_njop_bng;
17    @Getter @Setter @NonNull
18    private String op_wp_id;
```

2.19 Kelas `DatSubjekPajakModif`

Kelas ini pun digunakan untuk mengemas data dari tabel `dat_subjek_pajak` agar sesuai dengan kebutuhan bagian-depan (*frontend*), berikut adalah kodenya :

```
1     @Getter @Setter @NonNull
2     private String wp_nama;
3     @Getter @Setter @NonNull
4     private String wp_jalan;
5     @Getter @Setter @NonNull
6     private String wp_rtrw;
7     @Getter @Setter @NonNull
8     private String wp_kel;
9     @Getter @Setter @NonNull
10    private String wp_kota;
```

2.20 Kelas `PiutangSpptModif`

Kelas ini digunakan untuk mengemas informasi piutang agar sesuai dengan kebutuhan bagian-depan (*frontend*). Berikut adalah kode programnya :

```
1     @Getter @Setter
2     private String tgl_tempo;
3     @Getter @Setter
4     private BigDecimal jumlah;
5     @Getter @Setter
6     private String tgl_skrg;
7     @Getter @Setter
8     private int denda_bln;
9     @Getter @Setter
10    private BigDecimal denda_jml;
```

2.21 *Interface* `ReturnData`

Interface ini digunakan sebagai penyeragam nilai kembalian pada masing-masing titik API (*Application Programmable Interface*), sehingga nilai kembaliannya dapat lebih umum untuk diterapkan, deklarasinya sangat sederhana tanpa properti atau *method* apapun. Berikut adalah kodenya :

```
1 public interface ReturnData {}
```

2.22 Kelas `SpptModif`

Kelas ini digunakan untuk memodifikasi jumlah data yang akan dikirimkan ke bagian-depan (*frontend*) dari tabel `sppt` agar sesuai dengan kebutuhan bagian-depan (*frontend*). Kodenya adalah seperti berikut ini :

```
1     @Getter @Setter @NonNull
2     private String tahun;
```

```
3     @Getter @Setter @NonNull
4     private BigDecimal jumlah;
5     @Getter @Setter @NonNull
6     private String status;
```

3 BAGIAN-DEPAN (*FRONTEND*)

Bagian-depan (*frontend*) menggunakan Angular 5, sehingga banyak berkas yang terbentuk secara otomatis agar sistem berjalan sebagaimana mestinya. Beberapa penyesuaian dan penambahan modul atau berkas yang diperlukan adalah seperti berikut :

3.1 Berkas `app.component.html`

Berkas ini yang menjadi tampilan utama dan tampilan awal dari aplikasi sistem informasi pembayaran dari bagian-depan (*frontend*) ini. Berikut adalah kodenya :

```
1 <!--The content below is only a placeholder and can be replaced.-->
2 <header>
3   <nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
4     <a class="navbar-brand" href="#">BPPKAD Kab. Brebes</a>
5     <div class="collapse navbar-collapse" id="navbarCollapse">
6       <ul class="navbar-nav mr-auto"></ul>
7       <form class="form-inline mt-2 mt-md-0">
8         <input class="form-control mr-sm-2" name="nop" type="text"
    placeholder="NOP" aria-label="NOP" [(ngModel)]="nop"
9             [ngModelOptions]="{standalone: true}" #nopStatus="ngModel
    "
10            *ngIf="showNopTxt">
11        <button class="btn btn-outline-success my-2 my-sm-0" (click)=
    "cekNop(nop)" *ngIf="showCariBtn">Cari</button>
```

```
12          <button class="btn btn-outline-success my-2 my-sm-0" (click)=
    "kembali()" *ngIf="showKembaliBtn">
13              Cari NOP Lain</button>
14        </form>
15      </div>
16    </nav>
17 </header>
18 <main role="main" class="container">
19   <div class="mt-3">
20     <router-outlet></router-outlet>
21   </div>
22 </main>
```

Bagian masukkan teks untuk Nomor Objek Pajak (NOP) akan *binding* dengan variabel nop pada kelas `AppComponent` pada berkas `app.component.ts`.

Tombol `Cari` dan `Cari NOP Lain` akan berganti secara otomatis menyesuaikan kondisi properti `showCariBtn` dan `showKembaliBtn` milik kelas `AppComponent`.

Kemudian bagian penting dari halaman ini adalah adanya komponen `<router-outlet>` yang fungsinya adalah sebagai bagian dari tampilan yang secara dinamis akan diubah dengan tampilan lain melalui tombol `Cari`. Bagian *router* ini akan berkaitan erat dengan konfigurasi pada berkas `app-routing.module.ts`.

3.2 Berkas `app.component.ts`

Berkas ini berisi deklarasi kelas `AppComponent` yang akan beriringan dengan berkas `HTML` bernama `app.component.html` seperti pada deklarasinya berikut :

```
1 @Component({
2   selector: 'app-root',
3   templateUrl: './app.component.html',
4   styleUrls: ['./app.component.css']
5 })
```

Beberapa deklarasi properti yang ada pada kelas ini adalah sebagai berikut :

```
1  nop: string;
2  showNopTxt: boolean = true;
3  showCariBtn: boolean = true;
4  showKembaliBtn: boolean = false;
```

Deklarasi konstruktor pada kelas ini sederhana, hanya memeriksa apakah Nomor Objek Pajak (NOP) telah terisi atau belum, bila sudah terisi maka periksa Nomor Objek Pajak (NOP) tersebut. Sekaligus pada deklarasi konstruktor ini adalah melakukan pembentukan instan dari kelas-kelas yang dibutuhkan pada kelas `AppComponent` ini yang disebutkan dalam parameter pembentukan kontruktor ini, berikut adalah kode programnya :

```
1  constructor(private router: Router, private zone: NgZone) {
2    if(this.nop != null)
3      this.cekNop(this.nop);
4  }
```

Yang berikutnya adalah fungsi atau *method* dari `cekNop` dengan parameter Nomor Objek Pajak (NOP) yang nantinya akan melakukan navigasi ke alamat `/sppt/` yang didefinisikan pada berkas `app-routing.module.ts`. Berikut adalah isi kode dari fungsi `cekNop()` ini :

```
1  public cekNop(nop) {
2    if(nop.length != 18) {
3      return;
4    }
5    this.zone.run(() => {
6      this.showNopTxt = false;
7      this.showCariBtn = false;
8      this.showKembaliBtn = true;
9      this.router.navigate(['/sppt/' + nop]);
10   });
```

```
11   };
```

3.3 Berkas `app-routing.module.ts`

Kelas ini berisi deklarasi untuk *routing* perubahan sebagian halaman pada berkas `app.component.html`, tepatnya pada *tag* `<router-outlet>`. Berikut adalah deklarasi kelas dari `AppRoutingModule` ini :

```
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { HomeComponent } from './home/home.component';
4 import { SpptComponent } from './sppt/sppt.component';
5
6 const routes: Routes = [
7   { path: '', redirectTo: '/home', pathMatch: 'full' },
8   { path: 'home', component: HomeComponent },
9   { path: 'sppt/:nop', component: SpptComponent }
10 ];
11
12 @NgModule({
13   imports: [ RouterModule.forRoot(routes) ],
14   exports: [ RouterModule ],
15   declarations: []
16 })
17 export class AppRoutingModule { }
```

Terlihat bahwa pada saat *routing* melakukan navigasi ke *path* `sppt/:nop`, dimana `:nop` akan terisi oleh Nomor Objek Pajak (NOP), maka komponen `router-outlet` akan terisi oleh `SpptComponent`.

3.4 Berkas `sppt.component.html`

Berkas inilah yang akan tampil saat Nomor Objek Pajak telah terisi dan tombol `Cari` ditekan. Kodenya adalah seperti berikut :

```
<div class="tinggi">
  <div class="card">
    <div class="card-header">
      Data Objek Pajak
    </div>
    <div class="card-body">
<table>
  <tr>
    <td>NOP</td>
    <td>:</td>
    <td>{{ nop }}</td>
  </tr>
  <tr>
    <td>Luas Bumi</td>
    <td>:</td>
    <td>{{ luasBumi | number }} m<sup>2</sup></td>
  </tr>
  <tr>
    <td>Luas Bangunan</td>
    <td>:</td>
    <td>{{ luasBangunan | number }} m<sup>2</sup></td>
  </tr>
  <tr>
    <td>NJOP Bumi</td>
    <td>:</td>
    <td>Rp. {{ njopBumi | number }}.-</td>
  </tr>
  <tr>
    <td>NJOP Bangunan</td>
```

```
30      <td>:</td>
31      <td>Rp. {{ njopBangunan | number }}.-</td>
32    </tr>
33    <tr>
34      <td>Lokasi</td>
35      <td>:</td>
36      <td>{{ lokasi }}</td>
37    </tr>
38  </table>
39      </div>
40
41    </div>
42
43    <div style="padding-top: 20px;"></div>
44    <div class="card">
45      <div class="card-header">
46        Data Subjek Pajak
47      </div>
48      <div class="card-body">
49        <table>
50          <tr>
51            <td>Subjek Pajak ID</td>
52            <td>:</td>
53            <td>{{ spId }}</td>
54          </tr>
55          <tr>
56            <td>Nama Subjek Pajak</td>
57            <td>:</td>
58            <td>{{ namaSp }}</td>
59          </tr>
60          <tr>
61            <td>Alamat Subjek Pajak</td>
62            <td>:</td>
```

```
63            <td>{{ alamatSp }}</td>
64          </tr>
65        </table>
66      </div>
67    </div>
68
69    <div style="padding-top: 20px;"></div>
70    <div class="card">
71        <div class="card-header">
72          Data SPPT
73        </div>
74        <div class="card-body">
75          <table>
76            <tr *ngFor="let sppt of listSppt">
77              <td>Tahun {{ sppt.tahun }}</td>
78              <td>:</td>
79              <td>Rp. {{ sppt.jumlah | number }}.-</td>
80              <td>{{ sppt.status }}</td>
81            </tr>
82          </table>
83        </div>
84    </div>
85 </div>
```

3.5 Berkas `sppt.component.ts`

Berkas ini berisi deklarasi kelas `SpptComponent` yang merupakan *binding* kelas untuk halaman `sppt.component.html`. Beberapa properti yang ada pada kelas ini adalah seperti berikut :

```
1  nop: string;
2  luasBumi: number;
3  luasBangunan: number;
```

```
4   njopBumi: number;
5   njopBangunan: number;
6   lokasi: string;
7   spId: string;
8   namaSp: string;
9   alamatSp: string;
10  listSppt: any;
11  apiUrl = 'http://pajak-daerah.brebeskab.go.id:1328/android_service'
```

Konstruktor dari kelas ini hanya berisi pembentukan instan kelas yang akan dibutuhkan seperti deklarasinya dalam parameter konstruktor berikut :

```
1   constructor(
2     private route: ActivatedRoute ,
3     private http: HttpClient,
4     private zone: NgZone
5   ) { }
```

Karena kelas ini melakukan implementasi `OnInit`, maka tiap pertama kali kelas ini terbentuk atau aktif kembali dari kondisi *refresh*, maka *method* `ngOnInit` akan selalu dijalankan yang isinya adalah seperti berikut :

```
1   ngOnInit() {
2     this.nop = this.route.snapshot.paramMap.get('nop');
3     let body = 'keyword=op&nop=' + this.nop;
4
5     this.zone.run(() => {
6     this.http.post(this.apiUrl, body)
7       .subscribe(data => {
8         this.luasBumi = data['op_luas_bumi'];
9         this.luasBangunan = data['op_luas_bng'];
10        this.njopBumi = data['op_njop_bumi'];
11        this.njopBangunan = data['op_njop_bng'];
12        this.lokasi = data['op_jalan'] + ' ' + data['op_rtrw'] + ' '
    + data['op_kel'] + ', ' + data['op_kec'];
```

```
13          this.spId = data['op_wp_id'];
14          this.initSp(this.spId);
15        });
16
17        let spptBody = 'keyword=sppt&nop=' + this.nop;
18        this.http.post(this.apiUrl + '/sppt', spptBody)
19        .subscribe(data => {
20          this.listSppt = data;
21          console.log(data);
22        });
23      });
24    }
```

Isinya adalah melakukan *request* atau permintaan data ke bagian-belakang (*backend*) dengan `keyword op` dan `sppt`.

Di dalamnya terdapat pemanggilan ke fungsi atau *method* `initSp()` yang akan melakukan *request* atau permintaan data dengan `keyword wp`. Berikut adalah deklarasi dari *method* `initSp()` ini :

```
1   initSp(id): void {
2     let bodySp = 'keyword=wp&subjek_pajak_id=' + this.spId;
3     this.http.post(this.apiUrl, bodySp)
4       .subscribe(data => {
5         this.namaSp = data['wp_nama'];
6         this.alamatSp = data['wp_jalan'] + ' ' + data['wp_rtrw'] + '
    ' + data['wp_kel'] + ', ' + data['wp_kota'];
7       });
8   }
```

3.6 Berkas `home.component.html`

Berkas ini berisi tampilan awal dari bagian `routing-outlet` sebelum terjadinya proses pencarian sebuah objek pajak berdasarkan Nomor Objek Pajak (NOP).

Berikut adalah isi kode programnya :

```
<div class="tinggi">

<div class="card">
  <div class="card-body">
    <div class="card-title">
      <h1>Selamat Datang di Aplikasi WEB BPPKAD Kab. Brebes</h1>
    </div>
    <div class="card-text">
<p>Silahkan ketikan Nomor Objek Pajak (NOP) pada kotak kecil di kanan
    atas untuk mengetahui
  informasi PBB-P2 Anda. Kami pun menyediakan aplikasi Android yang
    dapat Anda unduh
  <a href="https://play.google.com/store/apps/details?id=com.
    brebeskab.mpbb">disini</a>
</p>
</div>
</div>
</div>
</div>
```

3.7 Kelas `home.component.ts`

Berkas ini berisi kelas `HomeComponent` yang merupakan *binding* dari halaman `home.component.html` yang didalamnya tidak perlu ada properti atau *method* yang terpasang, karena hanya menampilkan informasi saja. Berikut adalah isi kode programnya :

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-home',
```

```
5   templateUrl: './home.component.html',
6   styleUrls: ['./home.component.css']
7 })
8 export class HomeComponent implements OnInit {
9
10  constructor() { }
11
12  ngOnInit() {}
13 }
```