SOURCE CODE PROGRAM *WEB SERVICE* PBB-P2.

25 November 2016

Priyanto Tamami, S.Kom.

1 *SOURCE CODE* PEMBUATAN TABEL

Source code untuk pembuatan tabel menggunakan bahasa SQL biasa, adapun tabel yang digunakan ada 4 (empat) buah dengan rincian sebagai berikut :

1.1 Tabel SPPT

Tabel ini sudah ada terlebih dahulu sebagai tempat untuk menampung seluruh ketetapan yang ada pada PBB-P2, *source code* dari pembuatan tabel SPPT ini adalah sebagai berikut :

```
1 CREATE TABLE SPPT (
2   KD_PROPINSI CHAR(2) NOT NULL,
3   KD_DATI2 CHAR(2) NOT NULL,
4   KD_KECAMATAN CHAR(3) NOT NULL,
5   KD_KELURAHAN CHAR(3) NOT NULL,
6   KD_BLOK CHAR(3) NOT NULL,
7   NO_URUT CHAR(4) NOT NULL,
8   KD_JNS_OP CHAR(1) NOT NULL,
9   THN_PAJAK_SPPT CHAR(4) NOT NULL,
10  SIKLUS_SPPT NUMBER(2,0) NOT NULL,
11  KD_KANWIL_BANK CHAR(2) NOT NULL,
12  KD_KPPBB_BANK CHAR(2) NOT NULL,
13  KD_BANK_TUNGGAL CHAR(2) NOT NULL,
14  KD_BANK_PERSEPSI CHAR(2) NOT NULL,
15  KD_TP CHAR(2) NOT NULL,
```

```
16  NM_WP_SPPT VARCHAR2(30) NOT NULL,
17  JLN_WP_SPPT VARCHAR2(30) NOT NULL,
18  BLOK_KAV_NO_WP_SPPT VARCHAR2(30),
19  RW_WP_SPPT CHAR(2),
20  RT_WP_SPPT CHAR(3),
21  KELURAHAN_WP_SPPT VARCHAR2(30),
22  KOTA_WP_SPPT VARCHAR2(30),
23  KD_POS_WP_SPPT VARCHAR2(5),
24  NPWP_SPPT VARCHAR2(15),
25  NO_PERSIL_SPPT VARCHAR2(5),
26  KD_KLS_TANAH CHAR(3) DEFAULT 'XXX' NOT NULL,
27  THN_AWAL_KLS_TANAH CHAR(4) DEFAULT '1986' NOT NULL,
28  KD_KLS_BNG CHAR(3) DEFAULT 'XXX' NOT NULL,
29  THN_AWAL_KLS_BNG CHAR(4) DEFAULT '1986' NOT NULL,
30  TGL_JATUH_TEMPO_SPPT DATE NOT NULL,
31  LUAS_BUMI_SPPT NUMBER(12,0) DEFAULT 0 NOT NULL,
32  LUAS_BNG_SPPT NUMBER(12,0) DEFAULT 0 NOT NULL,
33  NJOP_BUMI_SPPT NUMBER(15,0) DEFAULT 0 NOT NULL,
34  NJOP_BNG_SPPT NUMBER(15,0) DEFAULT 0 NOT NULL,
35  NJOP_SPPT NUMBER(15,0) NOT NULL,
36  NJOPTKP_SPPT NUMBER(8,0) NOT NULL,
37  NJKP_SPPT NUMBER(5,2),
38  PBB_TERHUTANG_SPPT NUMBER(15,0) NOT NULL,
39  FAKTOR_PENGURANG_SPPT NUMBER(12,0),
40  PBB_YG_HARUS_DIBAYAR_SPPT NUMBER(15,0) NOT NULL,
41  STATUS_PEMBAYARAN_SPPT CHAR(1) DEFAULT '0' NOT NULL,
42  STATUS_TAGIHAN_SPPT CHAR(1) DEFAULT '0' NOT NULL,
43  STATUS_CETAK_SPPT CHAR(1) DEFAULT '0' NOT NULL,
44  TGL_TERBIT_SPPT DATE NOT NULL,
45  TGL_CETAK_SPPT DATE DEFAULT SYSDATE NOT NULL,
46  NIP_PENCETAK_SPPT CHAR(9) NOT NULL,
47  CONSTRAINT PK_E6 PRIMARY KEY (KD_PROPINSI, KD_DATI2, KD_KECAMATAN,
      KD_KELURAHAN, KD_BLOK, NO_URUT, KD_JNS_OP, THN_PAJAK_SPPT)
```

```
48   FOREIGN KEY (NIP_PENCETAK_SPPT) REFERENCES PEGAWAI(NIP),
49   FOREIGN KEY (KD_PROPINSI, KD_DATI2, KD_KECAMATAN, KD_KELURAHAN,
       KD_BLOK, NO_URUT,          KD_JNS_OP) REFERENCES DAT_OBJEK_PAJAK(
       KD_PROPINSI, KD_DATI2, KD_KECAMATAN, KD_KELURAHAN, KD_BLOK,
       NO_URUT, KD_JNS_OP),
50   FOREIGN KEY (KD_KANWIL_BANK, KD_KPPBB_BANK, KD_BANK_TUNGGAL,
       KD_BANK_PERSEPSI, KD_TP) REFERENCES TEMPAT_PEMBAYARAN(KD_KANWIL,
       KD_KPPBB, KD_BANK_TUNGGAL, KD_BANK_PERSEPSI, KD_TP),
51   FOREIGN KEY(KD_KLS_BNG, THN_AWAL_KLS_BNG) REFERENCES KELAS_BANGUNAN
       (KD_KLS_BNG, THN_AWAL_KLS_BNG),
52   FOREIGN KEY(KD_KLS_TANAH, THN_AWAL_KLS_TANAH) REFERENCES
       KELAS_TANAH(KD_KLS_TANAH, THN_AWAL_KLS_TANAH)
53 );
54
```

1.2 Tabel PEMBAYARAN_SPPT

Tabel ini pun sudah ada pada basis data yang digunakan sebagai tempat menyimpan atau mencatat transaksi pembayaran. *Source code* dari pembuatan tabel ini adalah sebagai berikut :

```
1 CREATE TABLE PEMBAYARAN_SPPT (
2   KD_PROPINSI CHAR(2) NOT NULL,
3   KD_DATI2 CHAR(2) NOT NULL,
4   KD_KECAMATAN CHAR(3) NOT NULL,
5   KD_KELURAHAN CHAR(3) NOT NULL,
6   KD_BLOK CHAR(3) NOT NULL,
7   NO_URUT CHAR(4) NOT NULL,
8   KD_JNS_OP CHAR(1) NOT NULL,
9   THN_PAJAK_SPPT CHAR(4) NOT NULL,
10  PEMBAYARAN_SPPT_KE NUMBER(2,0) NOT NULL,
11  KD_KANWIL_BANK CHAR(2) NOT NULL,
12  KD_KPPBB_BANK CHAR(2) NOT NULL,
```

```
13   KD_BANK_TUNGGAL CHAR(2) NOT NULL,
14   KD_BANK_PERSEPSI CHAR(2) NOT NULL,
15   KD_TP CHAR(2) NOT NULL,
16   DENDA_SPPT NUMBER(12,0),
17   JML_SPPT_YG_DIBAYAR NUMBER(15,0) NOT NULL,
18   TGL_PEMBAYARAN_SPPT DATE NOT NULL,
19   TGL_REKAM_BYR_SPPT DATE DEFAULT SYSDATE NOT NULL,
20   NIP_REKAM_BYR_SPPT CHAR(9) NOT NULL,
21   CONSTRAINT PK_G1 PRIMARY KEY(KD_PROPINSI, KD_DATI2, KD_KECAMATAN,
       KD_KELURAHAN, KD_BLOK, NO_URUT, KD_JNS_OP, THN_PAJAK_SPPT,
       PEMBAYARAN_SPPT_KE, KD_KANWIL_BANK, KD_KPPBB_BANK, KD_BANK_TUNGGAL
       , KD_BANK_PERSEPSI, KD_TP),
22   FOREIGN KEY (NIP_REKAM_BYR_SPPT) REFERENCES PEGAWAI(NIP),
23   FOREIGN KEY (KD_PROPINSI, KD_DATI2, KD_KECAMATAN, KD_KELURAHAN,
       KD_BLOK, NO_URUT, KD_JNS_OP, THN_PAJAK_SPPT) REFERENCES SPPT(
       KD_PROPINSI, KD_DATI2, KD_KECAMATAN, KD_KELURAHAN, KD_BLOK,
       NO_URUT, KD_JNS_OP, THN_PAJAK_SPPT),
24   FOREIGN KEY (KD_KANWIL_BANK, KD_KPPBB_BANK, KD_BANK_TUNGGAL,
       KD_BANK_PERSEPSI, KD_TP) REFERENCES TEMPAT_PEMBAYARAN(KD_KANWIL,
       KD_KPPBB, KD_BANK_TUNGGAL, KD_BANK_PERSEPSI, KD_TP)
25 );
26
```

1.3 Tabel LOG_TRX_PEMBAYARAN

Tabel ini digunakan untuk menyimpan atau mencatat proses transaksi pembayaran yang berhasil dilakukan. *Source code* untuk tabel LOG_TRX_PEMBAYARAN ini adalah sebagai berikut :

```
1 CREATE TABLE LOG_TRX_PEMBAYARAN (
2   NOP VARCHAR2(18 BYTE) NOT NULL,
3   THN VARCHAR2(4 BYTE) NOT NULL,
4   NTPD VARCHAR2(30 BYTE) NOT NULL,
```

```
5   POKOK NUMBER,
6   NAMA_WP VARCHAR2(50 BYTE),
7   ALAMAT_OP VARCHAR2(150 BYTE),
8   MATA_ANGGARAN VARCHAR2(15 BYTE),
9   MA_SANKSI VARCHAR2(20 BYTE),
10  DENDA NUMBER,
11  PEMBAYARAN_KE NUMBER(2,0),
12  IP_CLIENT VARCHAR2(30 BYTE),
13  CONSTRAINT LOG_TRX_PEMBAYARAN_PK PRIMARY KEY
14     ("NOP", "THN", "NTPD"));
15
```

1.4 Tabel LOG_REVERSAL

Tabel ini digunakan untuk mencatat historis transaksi *reversal* pembayaran yang telah selesai dilakukan. *Source code* untuk tabel ini adalah sebagai berikut :

```
1 CREATE TABLE LOG_REVERSAL (
2   NOP VARCHAR2(20 BYTE) NOT NULL,
3   THN VARCHAR2(4 BYTE) NOT NULL,
4   NTPD VARCHAR2(30 BYTE) NOT NULL,
5   IP_CLIENT VARCHAR2(30 BYTE),
6   CONSTRAINT LOG_REVERSAL_PK PRIMARY KEY ("NOP", "THN", "NTPD"));
7
```

2 *SOURCE CODE STORE PROCEDURE*

Store procedure pada basis data akan terbagi menjadi 3 (tiga) bagian. Fungsi dari *store procedure* ini adalah sekumpulan baris program yang menggunakan bahasa PL/SQL yang didukung oleh sistem basis data Oracle, baris demi baris akan dijalankan oleh sistem basis data, yang memudahkan pemindahan data antar be-

berapa tabel terjadi dengan sangat cepat. Ketiga bagian *store procedure* yang digunakan pada sistem aplikasi ini adalah sebagai berikut :

2.1 *STORE PROCEDURE* SPPT_TERHUTANG

Store procedure ini akan bertugas melakukan *query* ke basis data untuk kepentingan menampilkan informasi tagihan PBB-P2. Kode untuk *store procedure* ini adalah sebagai berikut :

```
1  create or replace procedure sppt_terhutang(c_sppt out sys_refcursor,
       nop in varchar, thn in varchar) is
2    v_kdPropinsi sppt.kd_propinsi%type;
3    v_kdDati2 sppt.kd_dati2%type;
4    v_kdKecamatan sppt.kd_kecamatan%type;
5    v_kdKelurahan sppt.kd_kelurahan%type;
6    v_kdBlok sppt.kd_blok%type;
7    v_noUrut sppt.no_urut%type;
8    v_kdJnsOp sppt.kd_jns_op%type;
9  begin
10   v_kdPropinsi := substr(nop,1,2);
11   v_kdDati2 := substr(nop,3,2);
12   v_kdKecamatan := substr(nop,5,3);
13   v_kdKelurahan := substr(nop,8,3);
14   v_kdBlok := substr(nop,11,3);
15   v_noUrut := substr(nop,14,4);
16   v_kdJnsOp := substr(nop,18,1);
17
18   open c_sppt for
19     SELECT
20     (sppt.kd_propinsi
21     ||sppt.kd_dati2
22     ||sppt.kd_kecamatan
23     ||sppt.kd_kelurahan
```

```
24      ||sppt.kd_blok
25      ||sppt.no_urut
26      ||sppt.kd_jns_op)                AS NOP,
27      sppt.thn_pajak_sppt              AS THN,
28      sppt.nm_wp_sppt                  AS NAMA,
29      kel.nm_kelurahan||' - '||kec.nm_kecamatan as "ALAMAT_OP",
30      sppt.pbb_yg_harus_dibayar_sppt AS POKOK,
31      case
32        when thn > '2013' and sysdate between (tgl_jatuh_tempo_sppt+1)
      and add_months(tgl_jatuh_tempo_sppt, 1) then ceil(1 * 0.02 *
      pbb_yg_harus_dibayar_sppt)
33        when thn > '2013' and sysdate between (tgl_jatuh_tempo_sppt+1)
      and add_months(tgl_jatuh_tempo_sppt, 2) then ceil(2 * 0.02 *
      pbb_yg_harus_dibayar_sppt)
34        when thn > '2013' and sysdate between (tgl_jatuh_tempo_sppt+1)
      and add_months(tgl_jatuh_tempo_sppt, 3) then ceil(3 * 0.02 *
      pbb_yg_harus_dibayar_sppt)
35        when thn > '2013' and sysdate between (tgl_jatuh_tempo_sppt+1)
      and add_months(tgl_jatuh_tempo_sppt, 4) then ceil(4 * 0.02 *
      pbb_yg_harus_dibayar_sppt)
36        when thn > '2013' and sysdate between (tgl_jatuh_tempo_sppt+1)
      and add_months(tgl_jatuh_tempo_sppt, 5) then ceil(5 * 0.02 *
      pbb_yg_harus_dibayar_sppt)
37        when thn > '2013' and sysdate between (tgl_jatuh_tempo_sppt+1)
      and add_months(tgl_jatuh_tempo_sppt, 6) then ceil(6 * 0.02 *
      pbb_yg_harus_dibayar_sppt)
38        when thn > '2013' and sysdate between (tgl_jatuh_tempo_sppt+1)
      and add_months(tgl_jatuh_tempo_sppt, 7) then ceil(7 * 0.02 *
      pbb_yg_harus_dibayar_sppt)
39        when thn > '2013' and sysdate between (tgl_jatuh_tempo_sppt+1)
      and add_months(tgl_jatuh_tempo_sppt, 8) then ceil(8 * 0.02 *
      pbb_yg_harus_dibayar_sppt)
```

```
40      when thn > '2013' and sysdate between (tgl_jatuh_tempo_sppt+1)
    and add_months(tgl_jatuh_tempo_sppt, 9) then ceil(9 * 0.02 *
    pbb_yg_harus_dibayar_sppt)
41      when thn > '2013' and sysdate between (tgl_jatuh_tempo_sppt+1)
    and add_months(tgl_jatuh_tempo_sppt, 10) then ceil(10 * 0.02 *
    pbb_yg_harus_dibayar_sppt)
42      when thn > '2013' and sysdate between (tgl_jatuh_tempo_sppt+1)
    and add_months(tgl_jatuh_tempo_sppt, 11) then ceil(11 * 0.02 *
    pbb_yg_harus_dibayar_sppt)
43      when thn > '2013' and sysdate between (tgl_jatuh_tempo_sppt+1)
    and add_months(tgl_jatuh_tempo_sppt, 12) then ceil(12 * 0.02 *
    pbb_yg_harus_dibayar_sppt)
44      when thn > '2013' and sysdate between (tgl_jatuh_tempo_sppt+1)
    and add_months(tgl_jatuh_tempo_sppt, 13) then ceil(13 * 0.02 *
    pbb_yg_harus_dibayar_sppt)
45      when thn > '2013' and sysdate between (tgl_jatuh_tempo_sppt+1)
    and add_months(tgl_jatuh_tempo_sppt, 14) then ceil(14 * 0.02 *
    pbb_yg_harus_dibayar_sppt)
46      when thn > '2013' and sysdate between (tgl_jatuh_tempo_sppt+1)
    and add_months(tgl_jatuh_tempo_sppt, 15) then ceil(15 * 0.02 *
    pbb_yg_harus_dibayar_sppt)
47      when thn > '2013' and sysdate > add_months(tgl_jatuh_tempo_sppt
    , 15) then ceil(15 * 0.02 * pbb_yg_harus_dibayar_sppt)
48      else 0
49    end as denda,
50    sppt.status_pembayaran_sppt
51  FROM sppt
52  join ref_kecamatan kec on (
53     kec.kd_propinsi = sppt.kd_propinsi
54    and kec.kd_dati2  = sppt.kd_dati2
55    and kec.kd_kecamatan = sppt.kd_kecamatan)
56  join ref_kelurahan kel on (
57     kel.kd_propinsi = sppt.kd_propinsi
```

```
58      and kel.kd_dati2 = sppt.kd_dati2
59      and kel.kd_kecamatan = sppt.kd_kecamatan
60      and kel.kd_kelurahan = sppt.kd_kelurahan)
61    where
62      sppt.kd_propinsi = v_kdPropinsi and
63      sppt.kd_dati2 = v_kdDati2 and
64      sppt.kd_kecamatan = v_kdKecamatan and
65      sppt.kd_kelurahan = v_kdKelurahan and
66      sppt.kd_blok = v_kdBlok and
67      sppt.no_urut = v_noUrut and
68      sppt.kd_jns_op = v_kdJnsOp and
69      sppt.thn_pajak_sppt = thn;
70 end;
71
```

2.2 *STORE PROCEDURE* PROSES_PEMBAYARAN

Store procedure ini digunakan untuk melakukan pencatatan pembayaran. Kode untuk *store procedure* ini adalah sebagai berikut :

```
1  create or replace procedure proses_pembayaran(c_proses_pembayaran out
       sys_refcursor,
2      nop in varchar, thn in varchar, tgl_bayar in date, ip_client in
       varchar)
3  is
4    v_kdPropinsi sppt.kd_propinsi%type;
5    v_kdDati2 sppt.kd_dati2%type;
6    v_kdKecamatan sppt.kd_kecamatan%type;
7    v_kdKelurahan sppt.kd_kelurahan%type;
8    v_kdBlok sppt.kd_blok%type;
9    v_noUrut sppt.no_urut%type;
10   v_kdJnsOp sppt.kd_jns_op%type;
11   v_pokokTemp sppt.pbb_yg_harus_dibayar_sppt%type;
```

```
12   v_dendaTemp numeric;
13   v_tglJatuhTempo sppt.tgl_jatuh_tempo_sppt%type;
14   v_kdKanwilBank sppt.kd_kanwil_bank%type;
15   v_kdKppbbBank sppt.kd_kppbb_bank%type;
16   v_kdBankTunggal sppt.kd_bank_tunggal%type;
17   v_kdBankPersepsi sppt.kd_bank_persepsi%type;
18   v_kdTp sppt.kd_tp%type;
19   v_statusPembayaranSppt sppt.status_pembayaran_sppt%type;
20   v_pembayaranKe numeric;
21   v_tglRekam date;
22   v_jamRekam timestamp;
23
24   v_ntpd varchar2(50);
25   v_namaWp sppt.nm_wp_sppt%type;
26   v_panjangNamaWp number;
27   v_znt dat_op_bumi.kd_znt%type;
28   v_nilaiSistemBumi dat_op_bumi.nilai_sistem_bumi%type;
29   v_mataAnggaran varchar2(20); -- '4.1.1.11.01'; -- ref_jns_sektor.
       kd_sektor = 02
30                                -- '4.1.1.11.02'; -- ref_jns_sektor.
       kd_sektor = 01
31   v_mataAnggaranSanksi varchar2(20);
32   v_kdSektor ref_kelurahan.kd_sektor%type;
33   v_nmKelurahan ref_kelurahan.nm_kelurahan%type;
34   v_nmKecamatan ref_kecamatan.nm_kecamatan%type;
35
36 begin
37   -- error code in c_proses_pembayaran
38   -- 01 : NO DATA FOUND
39   -- 02 : POKOK PEMBAYARAN BERBEDA DENGAN TAGIHAN
40   -- 03 : SPPT SUDAH TERBAYAR
41   -- 04 : TAGIHAN SPPT DIBATALKAN
42
```

```
43  -- set parameter nop
44  v_kdPropinsi := substr(nop,1,2);
45  v_kdDati2 := substr(nop,3,2);
46  v_kdKecamatan := substr(nop,5,3);
47  v_kdKelurahan := substr(nop,8,3);
48  v_kdBlok := substr(nop,11,3);
49  v_noUrut := substr(nop,14,4);
50  v_kdJnsOp := substr(nop,18,1);
51
52  -- query data dari tabel sppt
53  select pbb_yg_harus_dibayar_sppt, tgl_jatuh_tempo_sppt,
      kd_kanwil_bank,
54    kd_kppbb_bank, kd_bank_tunggal, kd_bank_persepsi, kd_tp,
      status_pembayaran_sppt,
55    nm_wp_sppt
56  into v_pokokTemp, v_tglJatuhTempo, v_kdKanwilBank,
57    v_kdKppbbBank, v_kdBankTunggal, v_kdBankPersepsi, v_kdTp,
      v_statusPembayaranSppt,
58    v_namaWp
59  from sppt
60  where thn_pajak_sppt = thn
61    and kd_propinsi = v_kdPropinsi
62    and kd_dati2 = v_kdDati2
63    and kd_kecamatan = v_kdKecamatan
64    and kd_kelurahan = v_kdKelurahan
65    and kd_blok = v_kdBlok
66    and no_urut = v_noUrut
67    and kd_jns_op = v_kdJnsOp;
68
69  if(v_tglJatuhTempo = NULL) then
70    open c_proses_pembayaran for
71      select '01' as kode_error from dual; -- NO DATA FOUND
72    goto exit_karena_error;
```

```
73  end if;
74
75  -- get data from dat_op_bumi
76  select kd_znt, nilai_sistem_bumi
77  into v_znt, v_nilaiSistemBumi
78  from dat_op_bumi
79  where kd_propinsi = v_kdPropinsi
80    and kd_dati2 = v_kdDati2
81    and kd_kecamatan = v_kdKecamatan
82    and kd_kelurahan = v_kdKelurahan
83    and kd_blok = v_kdBlok
84    and no_urut = v_noUrut
85    and kd_jns_op = v_kdJnsOp;
86
87  -- ambil kode sektor dan nama kelurahan
88  select kd_sektor, nm_kelurahan
89  into v_kdSektor, v_nmKelurahan
90  from ref_kelurahan
91  where kd_propinsi = v_kdPropinsi
92    and kd_dati2 = v_kdDati2
93    and kd_kecamatan = v_kdKecamatan
94    and kd_kelurahan = v_kdKelurahan;
95
96  -- ambil nama kecamatan
97  select nm_kecamatan
98  into v_nmKecamatan
99  from ref_kecamatan
100 where kd_propinsi = v_kdPropinsi
101   and kd_dati2 = v_kdDati2
102   and kd_kecamatan = v_kdKecamatan;
103
104 -- cek pokok harus sama dengan pbb_yg_harus_dibayar_sppt
105 --if(v_pokokTemp != pokok) then
```

```
106  --  open c_proses_pembayaran for
107  --    select '02' as kode_error from dual; -- pokok pembayaran yang
       dibayarkan berbeda dengan sismiop
108  --  return;
109  --end if;
110  -- update: pembayaran langsung diambil dari db
111
112  -- bila status pembayaran 1 atau v_pokokTemp = 0, tidak perlu
       dibayarkan, sudah terbayar
113  if(v_statusPembayaranSppt = '1' or v_pokokTemp = 0) then
114    open c_proses_pembayaran for
115      select '03' as kode_error from dual;
116    goto exit_karena_error;
117  elsif(v_statusPembayaranSppt = 2) then
118  -- bila status pembayaran 2, tidak perlu dibayarkan, tagihan
       dibatalkan
119    open c_proses_pembayaran for
120      select '04' as kode_error from dual;
121    goto exit_karena_error;
122  end if;
123
124  -- cek denda
125  if(thn > '2013' and tgl_bayar between (v_tglJatuhTempo+1) and
       add_months(v_tglJatuhTempo,1)) then
126    v_dendaTemp := ceil(1 * 0.02 * v_pokokTemp);
127  elsif (thn > '2013' and tgl_bayar between (v_tglJatuhTempo+1) and
       add_months(v_tglJatuhTempo,2)) then
128    v_dendaTemp := ceil(2 * 0.02 * v_pokokTemp);
129  elsif (thn > '2013' and tgl_bayar between (v_tglJatuhTempo+1) and
       add_months(v_tglJatuhTempo,3)) then
130    v_dendaTemp := ceil(3 * 0.02 * v_pokokTemp);
131  elsif (thn > '2013' and tgl_bayar between (v_tglJatuhTempo+1) and
       add_months(v_tglJatuhTempo,4)) then
```

```
132     v_dendaTemp := ceil(4 * 0.02 * v_pokokTemp);
133   elsif (thn > '2013' and tgl_bayar between (v_tglJatuhTempo+1) and
       add_months(v_tglJatuhTempo,5)) then
134     v_dendaTemp := ceil(5 * 0.02 * v_pokokTemp);
135   elsif (thn > '2013' and tgl_bayar between (v_tglJatuhTempo+1) and
       add_months(v_tglJatuhTempo,6)) then
136     v_dendaTemp := ceil(6 * 0.02 * v_pokokTemp);
137   elsif (thn > '2013' and tgl_bayar between (v_tglJatuhTempo+1) and
       add_months(v_tglJatuhTempo,7)) then
138     v_dendaTemp := ceil(7 * 0.02 * v_pokokTemp);
139   elsif (thn > '2013' and tgl_bayar between (v_tglJatuhTempo+1) and
       add_months(v_tglJatuhTempo,8)) then
140     v_dendaTemp := ceil(8 * 0.02 * v_pokokTemp);
141   elsif (thn > '2013' and tgl_bayar between (v_tglJatuhTempo+1) and
       add_months(v_tglJatuhTempo,9)) then
142     v_dendaTemp := ceil(9 * 0.02 * v_pokokTemp);
143   elsif (thn > '2013' and tgl_bayar between (v_tglJatuhTempo+1) and
       add_months(v_tglJatuhTempo,10)) then
144     v_dendaTemp := ceil(10 * 0.02 * v_pokokTemp);
145   elsif (thn > '2013' and tgl_bayar between (v_tglJatuhTempo+1) and
       add_months(v_tglJatuhTempo,11)) then
146     v_dendaTemp := ceil(11 * 0.02 * v_pokokTemp);
147   elsif (thn > '2013' and tgl_bayar between (v_tglJatuhTempo+1) and
       add_months(v_tglJatuhTempo,12)) then
148     v_dendaTemp := ceil(12 * 0.02 * v_pokokTemp);
149   elsif (thn > '2013' and tgl_bayar between (v_tglJatuhTempo+1) and
       add_months(v_tglJatuhTempo,13)) then
150     v_dendaTemp := ceil(13 * 0.02 * v_pokokTemp);
151   elsif (thn > '2013' and tgl_bayar between (v_tglJatuhTempo+1) and
       add_months(v_tglJatuhTempo,14)) then
152     v_dendaTemp := ceil(14 * 0.02 * v_pokokTemp);
153   elsif (thn > '2013' and tgl_bayar between (v_tglJatuhTempo+1) and
       add_months(v_tglJatuhTempo,15)) then
```

```
154     v_dendaTemp := ceil(15 * 0.02 * v_pokokTemp);
155   elsif (thn > '2013' and tgl_bayar >= add_months(v_tglJatuhTempo,15)
      ) then
156     v_dendaTemp := ceil(15 * 0.02 * v_pokokTemp);
157   else v_dendaTemp := 0;
158   end if;
159
160   -- count pembayaran yang sudah masuk
161   select count(kd_propinsi) into v_pembayaranKe
162   from pembayaran_sppt
163   where thn_pajak_sppt = thn
164     and kd_propinsi = v_kdPropinsi
165     and kd_dati2 = v_kdDati2
166     and kd_kecamatan = v_kdKecamatan
167     and kd_kelurahan = v_kdKelurahan
168     and kd_blok = v_kdBlok
169     and no_urut = v_noUrut
170     and kd_jns_op = v_kdJnsOp;
171
172   if(v_pembayaranKe = null) then
173     v_pembayaranKe := 0;
174   end if;
175
176   v_tglRekam := sysdate;
177   v_jamRekam := current_timestamp;
178   v_pembayaranKe := v_pembayaranKe +1;
179
180   -- simpan pembayaran di tabel pembayaran_sppt
181   insert into pembayaran_sppt(kd_propinsi, kd_dati2, kd_kecamatan,
      kd_kelurahan,
182     kd_blok, no_urut, kd_jns_op, thn_pajak_sppt, pembayaran_sppt_ke,
183     kd_kanwil_bank, kd_kppbb_bank, kd_bank_tunggal, kd_bank_persepsi,
      kd_tp,
```

```
184     denda_sppt, jml_sppt_yg_dibayar, tgl_pembayaran_sppt,
185     tgl_rekam_byr_sppt, nip_rekam_byr_sppt)
186   values(v_kdPropinsi, v_kdDati2, v_kdKecamatan, v_kdKelurahan,
187     v_kdBlok, v_noUrut, v_kdJnsOp, thn, v_pembayaranKe,
188     v_kdKanwilBank, v_kdKppbbBank, v_kdBankTunggal, v_kdBankPersepsi,
        v_kdTp,
189     v_dendaTemp, (v_pokokTemp + v_dendaTemp), tgl_bayar,
190     v_tglRekam, '060000000');
191   commit;
192
193   -- task: susun ntpd
194   v_panjangNamaWp := length(v_namaWp);
195   v_ntpd := to_char(tgl_bayar,'YYYY') ||
196     substr(v_namaWp,-1,1) || substr(v_znt,1,1) || v_panjangNamaWp ||
197     substr(to_char(v_nilaiSistemBumi),1,1) ||
198     length(to_char(v_nilaiSistemBumi)) || to_char(v_jamRekam,'MI') ||
199     substr(v_namaWp,1,1) || substr(v_znt,-1,1) || to_char(tgl_bayar,'
      DD') ||
200     substr(to_char(v_nilaiSistemBumi),-1,1) ||
201     substr(to_char(v_pokokTemp),2,1) || to_char(v_jamRekam, 'HH24')
      ||
202     to_char(tgl_bayar,'MM');
203
204   -- respond untuk pembayaran yang sukses
205   if (v_kdsektor = '02') then
206     v_mataAnggaran := '4.1.1.11.01';
207   else
208     v_mataAnggaran := '4.1.1.11.02';
209   end if;
210
211   v_mataAnggaranSanksi := v_mataAnggaran;
212
213   -- SIMPAN TRANSAKSI DI LOG
```

```
214   INSERT INTO LOG_TRX_PEMBAYARAN
215     (NOP, THN, NTPD, MATA_ANGGARAN, POKOK, MA_SANKSI, DENDA, NAMA_WP,
        ALAMAT_OP, PEMBAYARAN_KE, ip_client)
216   VALUES(nop, thn, v_ntpd, v_mataAnggaran, v_pokokTemp,
       v_mataAnggaranSanksi, v_dendaTemp, v_namaWp, v_nmKelurahan || ' -
       ' ||v_nmKecamatan,
217      v_pembayaranKe, ip_Client);
218   commit;
219
220   open c_proses_pembayaran for
221     select nop as nop, thn as thn, v_ntpd as ntpd, v_mataAnggaran as
      mata_anggaran_pokok,
222       v_pokokTemp as pokok, v_mataAnggaranSanksi as
      mata_anggaran_sanksi, v_dendaTemp as sanksi,
223       v_namaWp as nama_wp, v_nmKelurahan || ' - ' || v_nmKecamatan as
       alamat_op
224     from dual;
225
226   <<exit_karena_error>>
227   return;
228 end;
229
```

2.3 *STORE PROCEDURE* REVERSAL_PEMBAYARAN

Store procedure ini akan bertugas melakukan pengembalian data ke kondisi sebelum terjadinya pembayaran. Kode dari *store procedure* ini adalah sebagai berikut :

```
1 create or replace procedure reversal_pembayaran(c_data out
    sys_refcursor,
```

```
2   v_nop in varchar, v_thn in varchar, v_ntpd in varchar, v_ip_client
      in varchar)
3 is
4   v_adaData numeric;
5   v_kdPropinsi sppt.kd_propinsi%type;
6   v_kdDati2 sppt.kd_dati2%type;
7   v_kdKecamatan sppt.kd_kecamatan%type;
8   v_kdKelurahan sppt.kd_kelurahan%type;
9   v_kdBlok sppt.kd_blok%type;
10  v_noUrut sppt.no_urut%type;
11  v_kdJnsOp sppt.kd_jns_op%type;
12  v_pembayaranKe log_trx_pembayaran.pembayaran_ke%type;
13
14 begin
15  -- error code in c_data
16  -- 01 : NO DATA FOUND
17  -- 02 : DATA GANDA
18
19  -- set parameter nop
20  v_kdPropinsi := substr(v_nop,1,2);
21  v_kdDati2 := substr(v_nop,3,2);
22  v_kdKecamatan := substr(v_nop,5,3);
23  v_kdKelurahan := substr(v_nop,8,3);
24  v_kdBlok := substr(v_nop,11,3);
25  v_noUrut := substr(v_nop,14,4);
26  v_kdJnsOp := substr(v_nop,18,1);
27
28  -- verifikasi data, ada atau ga
29  select count(log_trx.nop)
30  into v_adaData
31  from log_trx_pembayaran log_trx
32  where log_trx.ntpd = v_ntpd
33    and log_trx.nop = v_nop
```

```
34      and log_trx.thn = v_thn;
35
36    if(v_adaData = 0) then
37      open c_data for
38        select '01' as kode_error from dual;
39      goto exit_karena_error;
40    elsif(v_adaData > 1) then
41      open c_data for
42        select '02' as kode_error from dual;
43      goto exit_karena_error;
44    end if;
45
46    -- ambil data dari log_trx_pembayaran
47    select pembayaran_ke
48    into v_pembayaranKe
49    from log_trx_pembayaran log_trx
50    where log_trx.nop = v_nop
51      and log_trx.thn = v_thn
52      and log_trx.ntpd = v_ntpd;
53
54    -- hapus data di pembayaran_sppt
55    delete from pembayaran_sppt
56    where thn_pajak_sppt = v_thn
57      and kd_propinsi = v_kdPropinsi
58      and kd_dati2 = v_kdDati2
59      and kd_kecamatan = v_kdKecamatan
60      and kd_kelurahan = v_kdKelurahan
61      and kd_blok = v_kdBlok
62      and no_urut = v_noUrut
63      and kd_jns_op = v_kdJnsOp
64      and pembayaran_sppt_ke = v_pembayaranKe;
65    commit;
66
```

```
67   -- ubah isi sppt.status_pembayaran_sppt = '0'
68   update sppt
69   set status_pembayaran_sppt = '0'
70   where thn_pajak_sppt = v_thn
71     and kd_propinsi = v_kdPropinsi
72     and kd_dati2 = v_kdDati2
73     and kd_kecamatan = v_kdKecamatan
74     and kd_kelurahan = v_kdKelurahan
75     and kd_blok = v_kdBlok
76     and no_urut = v_noUrut
77     and kd_jns_op = v_kdJnsOp;
78   commit;
79
80   -- catat di log_reversal
81   insert into log_reversal log_r
82     (log_r.nop, log_r.thn, log_r.ntpd, log_r.ip_client)
83   values (v_nop, v_thn, v_ntpd, v_ip_client);
84   commit;
85
86   open c_data for
87     select v_nop as NOP, v_thn as THN, v_ntpd as NTPD from dual;
88
89   <<exit_karena_error>>
90   return;
91 end;
```

3 *SOURCE CODE BUILD TOOLS*

Untuk mempermudah pengelolaan ketergantungan pustaka, maka dibutuhkan *build tools*, *build tools* yang sudah teruji kehandalannya dan stabilitas, termasuk cakupan pustaka yang ada, maka maven adalah pilihannya. Berikut adalah kode

dari konfigurasi maven yang diperlukan :

```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
    www.w3.org/2001/XMLSchema-instance"
2   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.
    apache.org/maven-v4_0_0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4   <groupId>lab.aikibo</groupId>
5   <artifactId>learn-spring-rest-hibernate</artifactId>
6   <packaging>war</packaging>
7   <version>1.0-SNAPSHOT</version>
8   <name>learn-spring-rest-hibernate Maven Webapp</name>
9   <url>http://maven.apache.org</url>
10
11  <properties>
12    <spring.version>4.0.6.RELEASE</spring.version>
13    <hibernate.version>4.3.6.Final</hibernate.version>
14    <log4j.version>1.2.17</log4j.version>
15    <jackson.version>2.7.5</jackson.version>
16    <jdk.version>1.7</jdk.version>
17  </properties>
18
19  <dependencies>
20    <!-- unit testing -->
21    <dependency>
22      <groupId>junit</groupId>
23      <artifactId>junit</artifactId>
24      <version>3.8.1</version>
25      <scope>test</scope>
26    </dependency>
27
28    <!-- log4j -->
29    <dependency>
```

```
30        <groupId>log4j</groupId>
31        <artifactId>log4j</artifactId>
32        <version>${log4j.version}</version>
33    </dependency>
34
35    <!-- oracle ojdbc -->
36    <dependency>
37        <groupId>oracle</groupId>
38        <artifactId>ojdbc14</artifactId>
39        <version>10.2.0.4</version>
40    </dependency>
41
42    <!-- spring -->
43    <dependency>
44        <groupId>org.springframework</groupId>
45        <artifactId>spring-core</artifactId>
46        <version>${spring.version}</version>
47    </dependency>
48    <dependency>
49        <groupId>org.springframework</groupId>
50        <artifactId>spring-web</artifactId>
51        <version>${spring.version}</version>
52    </dependency>
53    <dependency>
54        <groupId>org.springframework</groupId>
55        <artifactId>spring-webmvc</artifactId>
56        <version>${spring.version}</version>
57    </dependency>
58    <dependency>
59        <groupId>org.springframework</groupId>
60        <artifactId>spring-tx</artifactId>
61        <version>${spring.version}</version>
62    </dependency>
```

```
63      <dependency>
64        <groupId>org.springframework</groupId>
65        <artifactId>spring-orm</artifactId>
66        <version>${spring.version}</version>
67      </dependency>
68      <dependency>
69        <groupId>org.springframework</groupId>
70        <artifactId>spring-context</artifactId>
71        <version>${spring.version}</version>
72      </dependency>
73
74      <!-- hibernate -->
75      <dependency>
76        <groupId>org.hibernate</groupId>
77        <artifactId>hibernate-core</artifactId>
78        <version>${hibernate.version}</version>
79      </dependency>
80
81      <!-- bonecp -->
82      <dependency>
83        <groupId>com.jolbox</groupId>
84        <artifactId>bonecp-provider</artifactId>
85        <version>0.8.0-alpha1</version>
86      </dependency>
87      <dependency>
88        <groupId>com.jolbox</groupId>
89        <artifactId>bonecp</artifactId>
90        <version>0.8.0.RELEASE</version>
91      </dependency>
92
93      <!-- hikaricp, pengganti bonecp -->
94      <dependency>
95        <groupId>com.zaxxer</groupId>
```

```
96        <artifactId>HikariCP</artifactId>
97        <version>2.4.7</version>
98      </dependency>
99
100     <!-- jsr303 validation -->
101     <dependency>
102       <groupId>javax.validation</groupId>
103       <artifactId>validation-api</artifactId>
104       <version>1.1.0.Final</version>
105     </dependency>
106     <dependency>
107       <groupId>org.hibernate</groupId>
108       <artifactId>hibernate-validator</artifactId>
109       <version>5.1.3.Final</version>
110     </dependency>
111
112     <!-- servlet + jsp + jstl -->
113     <dependency>
114       <groupId>javax.servlet</groupId>
115       <artifactId>javax.servlet-api</artifactId>
116       <version>3.1.0</version>
117     </dependency>
118     <dependency>
119       <groupId>javax.servlet.jsp</groupId>
120       <artifactId>javax.servlet.jsp-api</artifactId>
121       <version>2.3.1</version>
122     </dependency>
123     <dependency>
124       <groupId>javax.servlet</groupId>
125       <artifactId>jstl</artifactId>
126       <version>1.2</version>
127     </dependency>
128
```

```
129     <!-- testing dependencies -->
130     <dependency>
131       <groupId>org.springframework</groupId>
132       <artifactId>spring-test</artifactId>
133       <version>${spring.version}</version>
134     </dependency>
135
136     <!-- mocking -->
137     <dependency>
138       <groupId>org.mockito</groupId>
139       <artifactId>mockito-core</artifactId>
140       <version>2.2.9</version>
141     </dependency>
142
143     <!-- jackson for convert POJO to json format -->
144     <dependency>
145       <groupId>com.fasterxml.jackson.core</groupId>
146       <artifactId>jackson-databind</artifactId>
147       <version>${jackson.version}</version>
148     </dependency>
149
150     <!-- joda time -->
151     <dependency>
152       <groupId>joda-time</groupId>
153       <artifactId>joda-time</artifactId>
154       <version>2.9.4</version>
155     </dependency>
156     <dependency>
157       <groupId>junit</groupId>
158       <artifactId>junit</artifactId>
159       <version>4.12</version>
160       <scope>test</scope>
161     </dependency>
```

```
162   </dependencies>
163
164   <build>
165     <finalName>learn-spring-rest-hibernate</finalName>
166
167     <plugins>
168       <plugin>
169         <groupId>org.apache.maven.plugins</groupId>
170         <artifactId>maven-compiler-plugin</artifactId>
171         <version>3.2</version>
172         <configuration>
173           <source>${jdk.version}</source>
174           <target>${jdk.version}</target>
175         </configuration>
176       </plugin>
177       <plugin>
178         <groupId>org.apache.maven.plugins</groupId>
179         <artifactId>maven-war-plugin</artifactId>
180         <version>2.4</version>
181         <configuration>
182           <warSourceDirectory>src/main/webapp</warSourceDirectory>
183           <warName>Spring-Rest-Hibernate-Pbb</warName>
184           <failOnMissingWebXml>false</failOnMissingWebXml>
185         </configuration>
186       </plugin>
187     </plugins>
188   </build>
189 </project>
```

4 *SOURCE CODE* KONFIGURASI application.properties

File application.properties ini adalah *file* konfigurasi yang digunakan dalam aplikasi nantinya. Isi dari *file* ini adalah sebagai berikut :

```
1 jdbc.driverClassName = oracle.jdbc.driver.OracleDriver
2 jdbc.url = jdbc:oracle:thin:***/***@192.168.2.7:1521/SISMIOP
3 jdbc.username = PBB
4 jdbc.password = RAHASIAPBB
5
6 hibernate.dialect = org.hibernate.dialect.Oracle10gDialect
7 hibernate.show_sql = true
8 hibernate.format_sql = true
9
10 # for bonecp
11 providerClass = com.zaxxer.hikari.hibernate.HikariConnectionProvider
12 minIdle = 2
13 maxPool = 30
14 dataSourceClassName = oracle.jdbc.pool.OracleDataSource
15 url = jdbc:oracle:thin:***/***@192.168.2.7:1521/sismiopbck
16 username = ***
17 password = ***
18 implicitCache = true
```

5 *SOURCE CODE* KONFIGURASI log4j.properties

Kode pada *file* log4j.properties ini adalah untuk mengatur keluaran dari *log* atau pencatat aktivitas pada saat sistem aplikasi berjalan, apakah akan dikelu-

arkan ke monitor, atau dicatatkan dalam *file* tertentu. Isi dari kode pada *file* log4j.properties ini adalah sebagai berikut :

```
# Root logger option
#log4j.rootLogger=TRACE, DEBUG, stdout, file
log4j.rootLogger=DEBUG, stdout, file

log4j.logger.org.hibernate.SQL = debug
log4j.logger.org.hibernate.type=trace

# Redirect log messages to console
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss
    } %-5p %c{1}:%L - %m%n

# Redirect log messages to a log file, support file rolling.
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=/var/lib/tomcat8/logs/log4j-app.log
log4j.appender.file.MaxFileSize=5MB
log4j.appender.file.MaxBackupIndex=10
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss}
    %-5p %c{1}:%L - %m%n
```

6 *SOURCE CODE* AppConfig.java

Kode pada *file* AppConfig.java adalah kode berisi konfigurasi dari *framework* Spring. Isi dari kode tersebut adalah sebagai berikut :

```
package lab.aikibo.config;

```

```
3 import org.springframework.context.annotation.ComponentScan;
4 import org.springframework.context.annotation.Configuration;
5 import org.springframework.context.annotation.Bean;
6 import org.springframework.context.annotation.Import;
7 import org.springframework.web.servlet.config.annotation.EnableWebMvc
      ;
8
9 import lab.aikibo.model.Sppt;
10 import lab.aikibo.dao.SpptDao;
11 import lab.aikibo.dao.SpptDaoImpl;
12
13 @Configuration
14 @EnableWebMvc
15 @ComponentScan(basePackages = {"lab.aikibo"})
16 @Import({HibernateConfiguration.class})
17 public class AppConfig {
18
19 }
```

Kelas AppConfig ini memang kosong, tidak perlu dilakukan pengaturan lain karena pengaturan-pengaturan dasar berada pada tanda *annotation* di atas kelas.

7 *SOURCE CODE* HibernateConfiguration.java

Kelas HibernateConfiguration ini adalah tempat untuk konfigurasi *framework* Hibernate yang nantinya berkomunikasi langsung dengan basis data. Kode dari kelas HibernateConfiguration ini adalah sebagai berikut :

```
1 package lab.aikibo.config;
2
3 import java.util.Properties;
```

```
4
5 import javax.sql.DataSource;
6
7 import org.hibernate.SessionFactory;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.context.annotation.Bean;
10 import org.springframework.context.annotation.ComponentScan;
11 import org.springframework.context.annotation.Configuration;
12 import org.springframework.context.annotation.PropertySource;
13 import org.springframework.core.env.Environment;
14 import org.springframework.jdbc.datasource.DriverManagerDataSource;
15 import org.springframework.orm.hibernate4.HibernateTransactionManager
    ;
16 import org.springframework.orm.hibernate4.LocalSessionFactoryBean;
17 import org.springframework.transaction.annotation.
    EnableTransactionManagement;
18
19 import com.zaxxer.hikari.HikariDataSource;
20
21 import com.jolbox.bonecp.BoneCPDataSource;
22
23 /**
24  * seharusnya ditambahkan bonecp
25  *
26  * @author Tamami
27  */
28 @Configuration
29 @EnableTransactionManagement
30 @ComponentScan({"lab.aikibo.config"})
31 @PropertySource(value = {"classpath:application.properties"})
32 public class HibernateConfiguration {
33   @Autowired
34   private Environment environment;
```

```
35
36  @Bean
37  public LocalSessionFactoryBean sessionFactory() {
38    LocalSessionFactoryBean sessionFactory = new
     LocalSessionFactoryBean();
39    sessionFactory.setDataSource(boneCPDataSource());
40    sessionFactory.setPackagesToScan(new String[] {"lab.aikibo.model
     "});
41    sessionFactory.setHibernateProperties(hibernateProperties());
42    return sessionFactory;
43  }
44
45  @Bean
46  public BoneCPDataSource boneCPDataSource() {
47    BoneCPDataSource ds = new BoneCPDataSource();
48    ds.setDriverClass(environment.getRequiredProperty("jdbc.
     driverClassName"));
49    ds.setJdbcUrl(environment.getRequiredProperty("jdbc.url"));
50    ds.setUsername(environment.getRequiredProperty("jdbc.username"));
51    ds.setPassword(environment.getRequiredProperty("jdbc.password"));
52
53    ds.setIdleConnectionTestPeriodInMinutes(60);
54    ds.setIdleMaxAgeInMinutes(420);
55    ds.setMaxConnectionsPerPartition(30);
56    ds.setMaxConnectionsPerPartition(10);
57    ds.setPartitionCount(10);
58    ds.setAcquireIncrement(5);
59    ds.setStatementsCacheSize(100);
60    ds.setReleaseHelperThreads(3);
61    return ds;
62  }
63
64  @Bean
```

```
65  public DataSource dataSource() {
66    DriverManagerDataSource dataSource = new DriverManagerDataSource
      ();
67    dataSource.setDriverClassName(environment.getRequiredProperty("
      jdbc.driverClassName"));
68    dataSource.setUrl(environment.getRequiredProperty("jdbc.url"));
69    dataSource.setUsername(environment.getRequiredProperty("jdbc.
      username"));
70    dataSource.setPassword(environment.getRequiredProperty("jdbc.
      password"));
71
72    return dataSource;
73  }
74
75  private Properties hibernateProperties() {
76    Properties properties = new Properties();
77    properties.put("hibernate.dialect", environment.
      getRequiredProperty("hibernate.dialect"));
78    properties.put("hibernate.show_sql", environment.
      getRequiredProperty("hibernate.show_sql"));
79    properties.put("hibernate.format_sql", environment.
      getRequiredProperty("hibernate.format_sql"));
80
81    properties.put("hibernate.hbm2ddl.auto", "validate");
82
83    return properties;
84  }
85
86  @Bean
87  @Autowired
88  public HibernateTransactionManager transactionManager(
      SessionFactory sf) {
```

```
89      HibernateTransactionManager txManager = new
        HibernateTransactionManager();
90      txManager.setSessionFactory(sf);
91      return txManager;
92    }
93 }
```

8 *SOURCE CODE* SerialConstant.java

Kelas SerialConstant akan berisi kode untuk kelas yang bersifat Serializable, karena setiap kelas yang mengimplementasikan Serializable memerlukan identitas untuk instan yang terbentuk. Berikut adalah isi dari kode kelas SerialConstant :

```
1  package lab.aikibo.constant;
2
3  public class SerialConstant {
4
5    public static final long SERIAL_SPPT = 1L;
6    public static final long SERIAL_SPPT_PK = 2L;
7    public static final long SERIAL_SPPT_SISMIOP = 3L;
8    public static final long SERIAL_SPPT_SISMIOP_PK = 4L;
9    public static final long SERIAL_REF_KELURAHAN = 5L;
10   public static final long SERIAL_REF_KELURAHAN_PK = 6L;
11   public static final long SERIAL_REF_KECAMATAN = 7L;
12   public static final long SERIAL_REF_KECAMATAN_PK = 8L;
13   public static final long SERIAL_PEMBAYARAN_SPPT = 9L;
14   public static final long SERIAL_PEMBAYARAN_SPPT_PK = 10L;
15   public static final long SERIAL_REVERSAL_PEMBAYARAN = 11L;
16
17 }
```

9 *SOURCE CODE* StatusRespond.java

Kelas StatusRespond ini akan menampung informasi kode respon yang akan diberikan ke *client*, agar memudahkan sistem aplikasi melakukan respon dan mencegah redundansi baris kode yang sama berulang-ulang. Kode dari kelas StatusRespond ini adalah sebagai berikut :

```
1  package lab.aikibo.constant;
2
3  public class StatusRespond {
4
5    public static final int APPROVED = 1;
6    public static final int DATA_INQ_NIHIL = 10;
7    public static final int TAGIHAN_TELAH_TERBAYAR = 13;
8    public static final int JUMLAH_SETORAN_NIHIL = 03; // pbb = 0
       rupiah
9    public static final int DATABASE_ERROR = 04;
10   public static final int ERROR_PEMBUATAN_NTPD = 05;
11   public static final int JUMLAH_PEMBAYARAN_BUKAN_ANGKA = 31;
12   public static final int TGL_JAM_BAYAR_LD_TGL_JAM_KIRIM = 32;
13   public static final int TGL_KIRIM_REV_LD_1H_ORI = 33;
14   public static final int PROC_CODE_NOT_VALID = 34;
15   public static final int MASA_PAJAK_BUKAN_ANGKA = 35;
16   public static final int THN_PAJAK_BUKAN_ANGKA = 36;
17
18 }
```

10 *SOURCE CODE* SpptRestController.java

Kelas ini nantinya akan menangani seluruh *request* yang masuk dari *client*. Kode untuk kelas ini adalah sebagai berikut :

```
1 package lab.aikibo.controller;
```

```
2
3 import org.apache.log4j.Logger;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.web.bind.annotation.PathVariable;
7 import org.springframework.web.bind.annotation.RequestMapping;
8 import org.springframework.web.bind.annotation.RestController;
9 import org.springframework.web.bind.annotation.RequestMethod;
10
11 import java.math.BigInteger;
12
13 import javax.servlet.http.HttpServletRequest;
14
15 import lab.aikibo.constant.StatusRespond;
16 import lab.aikibo.model.Sppt;
17 import lab.aikibo.model.SpptJ;
18 import lab.aikibo.model.Status;
19 import lab.aikibo.model.StatusInq;
20 import lab.aikibo.model.StatusTrx;
21 import lab.aikibo.model.StatusRev;
22 import lab.aikibo.model.Message;
23 import lab.aikibo.services.SpptServices;
24 import lab.aikibo.services.PembayaranServices;
25 import lab.aikibo.services.ReversalServices;
26
27 import org.joda.time.DateTime;
28
29 /**
30  * Mapping Rest dengan pola berikut :
31  *
32  * /Spring-Rest-Hibernate-Pbb/sppt/{nop}/{thn}
33  *   untuk inquiry data PBB
34  *
```

```
35  * /Spring-Rest-Hibernate-Pbb/bayar/{nop}/{thn}/{tglBayar}/{jamBayar}
36  *    untuk melakukan pencatatan pembayaran, request ini akan secara
        otomatis mencatatkan pembayaran
37  *    ke tabel pembayaran_sppt, bila ada kesalahan, maka perlu request
        reversal
38  *    Format tglBayar: DDMMYYYY
39  *    Format jamBayar: HH24MI
40  *
41  * /Spring-Rest-Hibernate-Pbb/reversal/{nop}/{thn}/{ntpd}
42  *    melakukan reversal atas pencatatan pembayaran sebelumnya.
43  *
44  * @author: tamami
45  */
46 @RestController
47 public class SpptRestController {
48
49   @Autowired
50   SpptServices spptServices;
51
52   @Autowired
53   PembayaranServices pembayaranServices;
54
55   @Autowired
56   ReversalServices reversalServices;
57
58   static final Logger logger = Logger.getLogger(SpptRestController.
       class);
59
60   public static Logger getLogger() {
61     return logger;
62   }
63
64   @RequestMapping("/")
```

```
65  public String welcome() {
66    String info = "Selamat Datang<br>";
67    info += "gunakan perintah berikut:<br>";
68    info += "<dl>";
69    info += "<dt>sppt/{nop}/{thn}</dt>";
70    info += "<dd>untuk inquiry data per nop</dd>";
71    info += "<dt>bayar/{nop}/{thn}/{tglBayar}/{jamBayar}</dt>";
72    info += "<dd>untuk melakukan pembayaran, data akan otomatis
      tersimpan sebagai transaksi pembayaran apabila ini di-request</dd
      >";
73    info += "<dt>reversal/{nop}/{thn}/{ntpd}</dt>";
74    info += "<dd>untuk melakukan reversal pembayaran</dd>";
75    info += "</dl>";
76    return info;
77  }
78
79  // single inquiry
80  @RequestMapping(value="/sppt/{nop}/{thn}", method = RequestMethod.
      GET)
81  public StatusInq getDataSppt(@PathVariable("nop") String nop,
      @PathVariable("thn") String thnPajak,
82      HttpServletRequest request) {
83    logger.debug("NOP: " + nop);
84    logger.debug("THN: " + thnPajak);
85    String ipClient = request.getHeader("X-FORWARDED-FOR");
86    if(ipClient == null) {
87      ipClient = request.getRemoteAddr();
88    }
89    StatusInq status = null;
90
91    // test thnPajak
92    try {
93      Integer.parseInt(thnPajak);
```

```
94      } catch(NumberFormatException ex) {
95        status = new StatusInq(StatusRespond.THN_PAJAK_BUKAN_ANGKA, "
        Tahun Pajak Mengandung Karakter bukan Angka", null);
96        return status;
97      }
98
99      try {
100       status = spptServices.getSpptByNopThn(nop, thnPajak, ipClient);
101     } catch(Exception e) {
102       logger.error(e);
103     }
104     logger.debug(" >>> GetData >>> " + status);
105     return status;
106   }
107
108   // single transaction
109   // format tanggal : DDMMYYYY
110   // format jam : HH24MI
111   @RequestMapping(value="/bayar/{nop}/{thn}/{tglBayar}/{jamBayar}",
        method = RequestMethod.GET)
112   public StatusTrx prosesPembayaran(@PathVariable("nop") String nop,
113       @PathVariable("thn") String thnPajak, @PathVariable("tglBayar")
        String tglBayarString,
114       @PathVariable("jamBayar")String jamBayarString,
        HttpServletRequest request) {
115     StatusTrx status = null;
116     BigInteger pokok = null;
117     BigInteger denda = null;
118     String ipClient = request.getHeader("X-FORWARDED-FOR");
119     if(ipClient == null) {
120       ipClient = request.getRemoteAddr();
121     }
122     logger.debug(" >>> IP CLIENT: " + ipClient);
```

```
123
124     // cek tanggal bayar, tidak boleh lebih baru daripada tanggal
      saat ini
125     int date = Integer.parseInt(tglBayarString.substring(0,2));
126     int month = Integer.parseInt(tglBayarString.substring(2,4));
127     int year = Integer.parseInt(tglBayarString.substring(4,8));
128     int hour = Integer.parseInt(jamBayarString.substring(0,2));
129     int min = Integer.parseInt(jamBayarString.substring(2,4));
130
131     DateTime tglBayar = new DateTime(year, month, date, hour, min);
132     if(tglBayar.isAfterNow()) {
133       // keluarkan pesan error
134       status = new StatusTrx(StatusRespond.
      TGL_JAM_BAYAR_LD_TGL_JAM_KIRIM,
135           "Tanggal atau jam pada saat dibayarkan melebihi tanggal dan
       jam saat ini", null);
136       return status;
137     }
138
139     // proses pembayaran
140     try {
141       status = pembayaranServices.prosesPembayaran(nop, thnPajak,
      tglBayar, ipClient);
142     } catch(Exception e) {
143       logger.error(e);
144       logger.debug(" >>> GetData >>> " + status);
145     }
146
147     return status;
148   }
149
150   @RequestMapping(value="/reversal/{nop}/{thn}/{ntpd}", method =
       RequestMethod.GET)
```

```
151   public StatusRev prosesReversal(@PathVariable("nop") String nop,
152       @PathVariable("thn") String thn, @PathVariable("ntpd") String
      ntpd,
153       HttpServletRequest request) {
154     StatusRev status = null;
155
156     // get IP Client
157     String ipClient = request.getHeader("X-FORWARDED-FOR");
158     if(ipClient == null) {
159       ipClient = request.getRemoteAddr();
160     }
161     logger.debug(" >>> IP CLIENT: " + ipClient);
162
163     try {
164       status = reversalServices.prosesReversal(nop, thn, ntpd,
      ipClient);
165     } catch(Exception ex) {
166       logger.error(ex);
167       logger.debug(" >>> GetData >>> " + status);
168     }
169     return status;
170   }
171
172   @RequestMapping(value="/info/{user}", method = RequestMethod.GET)
173   public Message getMessage(@PathVariable String user) {
174     Message msg = new Message(user, "Halo " + user);
175     return msg;
176   }
177
178 }
```

11 *SOUCE CODE* StoreProceduresDao.java

Kode StoreProceduresDao.java merupakan *interface* yang nantinya menjadi kerangka yang dibutuhkan *framework* Spring dalam mengorganisasi kelas-kelas yang nantinya akan melakukan komunikasi dengan basis data. Kode dari *interface* StoreProceduresDao adalah sebagai berikut :

```
1 package lab.aikibo.dao;
2
3 import lab.aikibo.model.StatusInq;
4 import lab.aikibo.model.StatusTrx;
5 import lab.aikibo.model.StatusRev;
6
7 import java.util.Date;
8
9 public interface StoreProceduresDao {
10   public StatusInq getDataSppt(String nop, String thn, String
       ipClient);
11   public StatusTrx prosesPembayaran(String nop, String thn, Date
       tglBayar, String ipClient);
12   public StatusRev reversalPembayaran(String nop, String thn, String
       ntpd, String ipClient);
13 }
```

12 *SOURCE CODE* StoreProceduresDaoImpl.java

Kode pada *file* StoreProceduresDaoImpl.java adalah kelas yang mengimplementasikan *interface* StoreProceduresDao yang nantinya akan melakukan komunikasi dengan sistem basis data. Kode dari kelas StoreProceduresDaoImpl adalah sebagai berikut :

```
1 package lab.aikibo.dao;
2
3 import org.springframework.stereotype.Repository;
4 import org.springframework.beans.factory.annotation.Autowired;
5
6 import java.util.List;
7
8 import java.sql.Connection;
9 import java.sql.CallableStatement;
10 import java.sql.ResultSet;
11 import java.sql.ResultSetMetaData;
12 import java.sql.SQLException;
13
14 import java.math.BigInteger;
15 import java.util.Date;
16
17 import javax.persistence.ParameterMode;
18
19 import org.hibernate.SessionFactory;
20 import org.hibernate.Query;
21 import org.hibernate.Session;
22 import org.hibernate.jdbc.Work;
23 import org.hibernate.procedure.ProcedureCall;
24 import org.hibernate.result.Output;
25 import org.hibernate.result.ResultSetOutput;
26
27 import org.joda.time.DateTime;
28 import org.joda.time.Months;
29
30 import oracle.jdbc.OracleTypes;
31
32 import com.jolbox.bonecp.BoneCPDataSource;
33
```

```
34 import lab.aikibo.model.Sppt;
35 import lab.aikibo.model.SpptJ;
36 import lab.aikibo.model.StatusInq;
37 import lab.aikibo.model.StatusRev;
38 import lab.aikibo.model.SpptSismiop;
39 import lab.aikibo.model.StatusTrx;
40 import lab.aikibo.model.PembayaranSppt;
41 import lab.aikibo.model.ReversalPembayaran;
42
43 import lab.aikibo.constant.StatusRespond;
44
45 import lab.aikibo.controller.SpptRestController;
46
47 @Repository("storeProceduresDao")
48 public class StoreProceduresDaoImpl implements StoreProceduresDao {
49
50   @Autowired
51   private SessionFactory sessionFactory;
52   @Autowired
53   private BoneCPDataSource boneCPDs;
54
55   CallableStatement callable;
56   Sppt sppt;
57   PembayaranSppt pembayaranSppt;
58   ReversalPembayaran revPembayaran;
59   StatusInq status;
60   StatusTrx statusTrx;
61   StatusRev statusRev;
62
63   public StatusInq getDataSppt(String nop, String thn, String
       ipClient) {
64     // -- ini cara 1; lumpuh saat panggil ke connection() deprecated
65
```

```
66      callable = null;
67      sppt = null;
68      status = null;
69
70      try {
71        callable = boneCPDs.getConnection().prepareCall("call
    sppt_terhutang(?,?,?)");
72        callable.registerOutParameter(1, OracleTypes.CURSOR);
73        callable.setString(2, nop);
74        callable.setString(3, thn);
75        //if(callable.executeUpdate() == 0) {
76        callable.executeUpdate();
77        ResultSet rs = (ResultSet) callable.getObject(1);
78        ResultSetMetaData rsmd = rs.getMetaData();
79        for(int i=1; i<rsmd.getColumnCount(); i++) {
80          SpptRestController.getLogger().debug(" >>> data " + i + " : "
     + rsmd.getColumnName(i));
81        }
82        sppt = new Sppt();
83
84        while(rs.next()) {
85          String nama = rs.getString("NAMA");
86          String alamatOp = rs.getString("ALAMAT_OP");
87          BigInteger pokok = rs.getBigDecimal("POKOK").toBigInteger();
88          BigInteger denda = rs.getBigDecimal("DENDA").toBigInteger();
89          SpptRestController.getLogger().debug(" >>> NAMA : " + nama);
90          SpptRestController.getLogger().debug(" >>> ALAMAT OP : " +
    alamatOp);
91          SpptRestController.getLogger().debug(" >>> POKOK : " + pokok)
    ;
92          SpptRestController.getLogger().debug(" >>> DENDA : " + denda)
    ;
93          sppt.setNop(nop);
```

```
94          sppt.setThn(thn);
95          sppt.setNama(nama);
96          sppt.setAlamatOp(alamatOp);
97          sppt.setPokok(pokok);
98          sppt.setDenda(denda);
99        }
100       if(sppt.getNop() == null) {
101         status = new StatusInq(StatusRespond.DATA_INQ_NIHIL, "Data
      Tidak Ditemukan", null);
102         return status;
103       }
104
105       status = new StatusInq(StatusRespond.APPROVED, "Data ditemukan
      ", sppt);
106       return status;
107     } catch(Exception e) {
108       status = new StatusInq(StatusRespond.DATABASE_ERROR, "Kesalahan
      DB", null);
109     }
110     return status;
111   }
112
113   public StatusTrx prosesPembayaran(String nop, String thn, Date
      tglBayar, String ipClient) {
114     callable = null;
115     pembayaranSppt = null;
116     statusTrx = null;
117
118     try {
119       callable = boneCPDs.getConnection().prepareCall("call
      proses_pembayaran(?,?,?,?,?)");
120       callable.registerOutParameter(1, OracleTypes.CURSOR);
121       callable.setString(2, nop);
```

```
122       callable.setString(3, thn);
123       callable.setDate(4, new java.sql.Date(tglBayar.getTime()));
124       callable.setString(5, ipClient);
125
126       callable.executeUpdate();
127       ResultSet rs = (ResultSet) callable.getObject(1);
128       ResultSetMetaData rsMeta = rs.getMetaData();
129       pembayaranSppt = new PembayaranSppt();
130       while(rs.next()) {
131         SpptRestController.getLogger().debug(" >>> Berhasil masuk
      iterasi rs.next");
132         SpptRestController.getLogger().debug(" >>> nama kolom yang
      ada : " + rsMeta.getColumnName(1));
133         if(!rsMeta.getColumnName(1).equals("KODE_ERROR")) {
134           SpptRestController.getLogger().debug(" >>> nop-nya ada : "
      + rs.getString("nop"));
135           pembayaranSppt.setNop(rs.getString("NOP"));
136           pembayaranSppt.setThn(rs.getString("THN"));
137           pembayaranSppt.setNtpd(rs.getString("NTPD"));
138           pembayaranSppt.setMataAnggaranPokok(rs.getString("
      MATA_ANGGARAN_POKOK"));
139           pembayaranSppt.setPokok(rs.getBigDecimal("POKOK").
      toBigInteger());
140           pembayaranSppt.setMataAnggaranSanksi(rs.getString("
      MATA_ANGGARAN_POKOK"));
141           pembayaranSppt.setSanksi(rs.getBigDecimal("SANKSI").
      toBigInteger());
142           pembayaranSppt.setNamaWp(rs.getString("NAMA_WP"));
143           pembayaranSppt.setAlamatOp(rs.getString("ALAMAT_OP"));
144         } else {
145           String infoSp = rs.getString("KODE_ERROR");
146           if(infoSp.equals("01")) {
```

```
147             statusTrx = new StatusTrx(StatusRespond.
    TAGIHAN_TELAH_TERBAYAR, "Tagihan Telah Terbayar atau Pokok Pajak
    Nihil.", null);
148             return statusTrx;
149           } else if(infoSp.equals("02")) {
150             // not used
151           } else if(infoSp.equals("03")) {
152             statusTrx = new StatusTrx(StatusRespond.
    TAGIHAN_TELAH_TERBAYAR, "Tagihan Telah Terbayar", null);
153             return statusTrx;
154           } else if(infoSp.equals("04")) {
155             statusTrx = new StatusTrx(StatusRespond.
    JUMLAH_SETORAN_NIHIL, "Tagihan SPPT Telah Dibatalkan", null);
156             return statusTrx;
157           }
158         }
159       }
160       statusTrx = new StatusTrx(StatusRespond.APPROVED, "Pembayaran
    Telah Tercatat", pembayaranSppt);
161     } catch(Exception ex) {
162       SpptRestController.getLogger().debug(" >>> hasil Exception : "
    + ex);
163       statusTrx = new StatusTrx(StatusRespond.DATABASE_ERROR, "
    Kesalahan Server", null);
164       return statusTrx;
165     }
166
167     return statusTrx;
168   }
169
170   public StatusRev reversalPembayaran(String nop, String thn, String
     ntpd, String ipClient) {
171     callable = null;
```

```
172     revPembayaran = null;
173     statusRev = null;
174 
175     try {
176       callable = boneCPDs.getConnection().prepareCall("call
    reversal_pembayaran(?,?,?,?,?)");
177       callable.registerOutParameter(1, OracleTypes.CURSOR);
178       callable.setString(2, nop);
179       callable.setString(3, thn);
180       callable.setString(4, ntpd);
181       callable.setString(5, ipClient);
182       callable.executeUpdate();
183 
184       ResultSet rs = (ResultSet) callable.getObject(1);
185       ResultSetMetaData rsMeta = rs.getMetaData();
186       ReversalPembayaran revBayar = new ReversalPembayaran();
187       while(rs.next()) {
188         if(!rsMeta.getColumnName(1).equals("KODE_ERROR")) {
189           revBayar.setNop(rs.getString("NOP"));
190           revBayar.setThn(rs.getString("THN"));
191           revBayar.setNtpd(rs.getString("NTPD"));
192           statusRev = new StatusRev(StatusRespond.APPROVED, "Proses
    Reversal Berhasil", revBayar);
193         } else {
194           String infoSp = rs.getString("KODE_ERROR");
195           if(infoSp.equals("01")) {
196             statusRev = new StatusRev(StatusRespond.DATA_INQ_NIHIL, "
    Data Yang Diminta Tidak Ada", null);
197             return statusRev;
198           } else if(infoSp.equals("02")) {
199             statusRev = new StatusRev(StatusRespond.DATABASE_ERROR, "
    Data tersebut Ganda", null);
200             return statusRev;
```

```
201             }
202           }
203         }
204       } catch(Exception ex) {
205         SpptRestController.getLogger().debug(" >>> hasil Exception : "
      + ex);
206         statusRev = new StatusRev(StatusRespond.DATABASE_ERROR, "
      Kesalahan Server", null);
207         return statusRev;
208       }
209       return statusRev;
210     }
211
212     private BigInteger hitungDenda(BigInteger pokok, Date tglJatuhTempo
      ) {
213       DateTime start = new DateTime(tglJatuhTempo);
214       DateTime end = new DateTime();
215       int selisih = selisihBulan(start, end);
216       if(selisih < 0) return new BigInteger("0");
217       else return new BigInteger("0");
218
219     }
220
221     public int selisihBulan(DateTime start, DateTime end) {
222       return Months.monthsBetween(start.withDayOfMonth(start.
      getDayOfMonth()),
223           end.withDayOfMonth(start.getDayOfMonth())).getMonths();
224     }
225
226 }
```

13 *SOURCE CODE* AppInitializer.java

File AppInitializer.java berisi kelas yang akan dipanggil atau dieksekusi pertama kali oleh *framework* Spring karena mengimplementasikan *interface* AbstractAnnotationConfigDispatcherServletInitializer. Isi dari kelas AppInitializer adalah sebagai berikut :

```
1  package lab.aikibo.init;
2
3  import org.springframework.web.servlet.support.
       AbstractAnnotationConfigDispatcherServletInitializer;
4
5  import lab.aikibo.config.AppConfig;
6
7  public class AppInitializer extends
       AbstractAnnotationConfigDispatcherServletInitializer {
8
9    @Override
10   protected Class<?>[] getRootConfigClasses() {
11     return new Class[] {AppConfig.class};
12   }
13
14   @Override
15   protected Class<?>[] getServletConfigClasses() {
16     return null;
17   }
18
19   @Override
20   protected String[] getServletMappings() {
21     return new String[] { "/" };
22   }
23
24 }
```

14 *SOURCE CODE* PembayaranSppt.java

File PembayaranSppt.java berisi kelas yang nantinya menampung data yang dikembalikan dari sistem basis data karena ada *request* pencatatan pembayaran dari *client*. Kode untuk kelas ini adalah sebagai berikut :

```
1  package lab.aikibo.model;
2
3  import java.io.Serializable;
4
5  import java.math.BigInteger;
6
7  import lab.aikibo.constant.SerialConstant;
8
9  public class PembayaranSppt implements Serializable {
10
11   private static final long serialVersionUID = SerialConstant.
       SERIAL_PEMBAYARAN_SPPT;
12
13   private String nop;
14   private String thn;
15   private String ntpd;
16   private String mataAnggaranPokok;
17   private BigInteger pokok;
18   private String mataAnggaranSanksi;
19   private BigInteger sanksi;
20   private String namaWp;
21   private String alamatOp;
22
23   // --- constructors
24
25   public PembayaranSppt() {}
26
```

```
27  public PembayaranSppt(String nop, String thn, String ntpd,
28      String mataAnggaranPokok, BigInteger pokok, String
     mataAnggaranSanksi,
29      BigInteger sanksi, String namaWp, String alamatOp) {
30    this.nop = nop;
31    this.thn = thn;
32    this.ntpd = ntpd;
33    this.mataAnggaranPokok = mataAnggaranPokok;
34    this.pokok = pokok;
35    this.mataAnggaranSanksi = mataAnggaranSanksi;
36    this.sanksi = sanksi;
37    this.namaWp = namaWp;
38    this.alamatOp = alamatOp;
39  }
40
41
42  // -- setter getter
43
44  public String getNop() { return nop; }
45
46  public void setNop(String nop) { this.nop = nop; }
47
48  public String getThn() { return thn; }
49
50  public void setThn(String thn) { this.thn = thn; }
51
52  public String getNtpd() { return ntpd; }
53
54  public void setNtpd(String ntpd) { this.ntpd = ntpd; }
55
56  public String getMataAnggaranPokok() { return mataAnggaranPokok; }
57
```

```
58    public void setMataAnggaranPokok(String mataAnggaranPokok) { this.
       mataAnggaranPokok = mataAnggaranPokok; }
59
60    public BigInteger getPokok() { return pokok; }
61
62    public void setPokok(BigInteger pokok) { this.pokok = pokok; }
63
64    public String getMataAnggaranSanksi() { return mataAnggaranSanksi;
       }
65
66    public void setMataAnggaranSanksi(String mataAnggaranSanksi) { this
       .mataAnggaranSanksi = mataAnggaranSanksi; }
67
68    public BigInteger getSanksi() { return sanksi; }
69
70    public void setSanksi(BigInteger sanksi) { this.sanksi = sanksi; }
71
72    public String getNamaWp() { return namaWp; }
73
74    public void setNamaWp(String namaWp) { this.namaWp = namaWp; }
75
76    public String getAlamatOp() { return alamatOp; }
77
78    public void setAlamatOp(String alamatOp) { this.alamatOp = alamatOp
       ; }
79
80 }
```

15 *SOURCE CODE* ReversalPembayaran.java

Isi dari *file* ReversalPembayaran.java adalah kelas ReversalPembayaran yang bertugas menampung data yang dikembalikan dari sistem basis data karena adanya

request reversal pembayaran dari *client*. Berikut adalah isi dari kodenya :

```
1 package lab.aikibo.model;
2
3 import java.io.Serializable;
4
5 import lab.aikibo.constant.SerialConstant;
6
7 public class ReversalPembayaran implements Serializable {
8
9   private static final long serialVersionUID = SerialConstant.
     SERIAL_REVERSAL_PEMBAYARAN;
10
11  private String nop;
12  private String thn;
13  private String ntpd;
14
15  public ReversalPembayaran() {};
16
17  public ReversalPembayaran(String nop, String thn, String ntpd) {
18    this.nop = nop;
19    this.thn = thn;
20    this.ntpd = ntpd;
21  }
22
23  // --- setter getter
24
25  public String getNop() { return nop; }
26
27  public void setNop(String nop) { this.nop = nop; }
28
29  public String getThn() { return thn; }
30
```

```
31    public void setThn(String thn) { this.thn = thn; }
32
33    public String getNtpd() { return ntpd; }
34
35    public void setNtpd(String ntpd) { this.ntpd = ntpd; }
36
37 }
```

16 *SOURCE CODE* Sppt.java

Isi dari *file* Sppt.java adalah kelas Sppt yang bertugas menyimpan data hasil kembalian dari sistem basis data untuk permintaan / *request inquiry* data tagihan PBB-P2. Baris kode dari kelas Sppt ini adalah sebagai berikut :

```
1  package lab.aikibo.model;
2
3  import java.io.Serializable;
4  import java.math.BigInteger;
5
6  import javax.persistence.Embeddable;
7  import javax.persistence.EmbeddedId;
8  import javax.persistence.Entity;
9  import javax.persistence.Table;
10 import javax.persistence.IdClass;
11 import javax.persistence.Id;
12 import javax.persistence.AttributeOverrides;
13 import javax.persistence.AttributeOverride;
14 import javax.persistence.Column;
15
16 import lab.aikibo.constant.SerialConstant;
17
18 @Entity
```

```
19 @Table(name="sppt_terhutang_matre")
20 @IdClass(Sppt.SpptPK.class)
21 public class Sppt implements Serializable {
22
23   private static final long serialVersionUID = SerialConstant.
       SERIAL_SPPT;
24
25   @Id
26   @Column(name="NOP", columnDefinition="char")
27   private String nop;
28   @Id
29   @Column(name="THN", columnDefinition="char")
30   private String thn;
31   @Column(name="NAMA")
32   private String nama;
33   @Column(name="ALAMAT_OP")
34   private String alamatOp;
35   @Column(name="POKOK")
36   private BigInteger pokok;
37   @Column(name="DENDA")
38   private BigInteger denda;
39
40
41
42   private String statusPembayaran;
43
44   // --- setter getter
45
46   public String getNop() {
47     return nop;
48   }
49
50   public void setNop(String nop) {
```

```
51      this.nop = nop;
52    }
53
54    public String getThn() {
55      return thn;
56    }
57
58    public void setThn(String thn) {
59      this.thn = thn;
60    }
61
62    public String getNama() {
63      return nama;
64    }
65
66    public void setNama(String nama) {
67      this.nama = nama;
68    }
69
70    public String getAlamatOp() {
71      return alamatOp;
72    }
73
74    public void setAlamatOp(String alamatOp) {
75      this.alamatOp = alamatOp;
76    }
77
78    public BigInteger getPokok() {
79      return pokok;
80    }
81
82    public void setPokok(BigInteger pokok) {
83      this.pokok = pokok;
```

```
84  }
85
86  public BigInteger getDenda() {
87    return denda;
88  }
89
90  public void setDenda(BigInteger denda) {
91    this.denda = denda;
92  }
93
94  public String getStatusPembayaran() {
95    return statusPembayaran;
96  }
97
98  public void setStatusPembayaran(String statusPembayaran) {
99    this.statusPembayaran = statusPembayaran;
100 }
101
102
103 // --- inner class
104 @Embeddable
105 public static class SpptPK implements Serializable {
106   private static final long serialVersionUID = SerialConstant.
      SERIAL_SPPT_PK;
107
108   protected String nop;
109   protected String thn;
110
111   public SpptPK() {}
112
113   public SpptPK(String nop, String thn) {
114     this.nop = nop;
115     this.thn = thn;
```

```
116     }
117
118     public String getNop() {
119       return nop;
120     }
121
122     public String getThn() {
123       return thn;
124     }
125
126     public void setNop(String nop) {
127       this.nop = nop;
128     }
129
130     public void setThn(String thn) {
131       this.thn = thn;
132     }
133
134   }
135
136 }
```

17 *SOURCE CODE* Status.java

Isi dari *file* Status.java adalah kelas Status yang merupakan kelas umum untuk menampilkan informasi ke *client* terhadap *request* yang diterima. Isi kode dari kelas Status ini adalah sebagai berikut :

```
1 package lab.aikibo.model;
2
3 public class Status {
4
```

```
5   private int code;
6   private String message;
7
8   public Status() {}
9
10  public Status(int code, String message) {
11    this.code = code;
12    this.message = message;
13  }
14
15  public int getCode() {
16    return code;
17  }
18
19  public String getMessage() {
20    return message;
21  }
22
23  public void setCode(int code) {
24    this.code = code;
25  }
26
27  public void setMessage(String message) {
28    this.message = message;
29  }
30
31 }
```

18 *SOURCE CODE* StatusInq.java

Isi dari *file* StatusInq.java ini adalah kelas StatusInq yang akan membungkus informasi untuk dikirimkan ke *client* terhadap *request inquiry* yang diterima. Kode

dari kelas StatusInq ini adalah sebagai berikut :

```
1 package lab.aikibo.model;
2
3 public class StatusInq extends Status {
4   private Sppt sppt;
5
6   public StatusInq() {}
7
8   public StatusInq(int code, String message, Sppt sppt) {
9     super(code, message);
10    this.sppt = sppt;
11  }
12
13  public Sppt getSppt() { return sppt; }
14
15  public void setSppt(Sppt sppt) { this.sppt = sppt; }
16
17 }
```

19 *SOURCE CODE* StatusRev.java

Isi dari *file* StatusRev.java adalah kelas StatusRev yang nantinya akan membungkus informasi yang akan dikirimkan ke *client* terhadap *request reversal* yang dikirimkan oleh *client* ke *server*. Kode dari kelas StatusRev ini adalah sebagai berikut :

```
1 package lab.aikibo.model;
2
3 public class StatusRev extends Status {
4   private ReversalPembayaran revPembayaran;
5
6   public StatusRev() {}
```

```
7
8   public StatusRev(int code, String message, ReversalPembayaran
      revPembayaran) {
9     super(code, message);
10    this.revPembayaran = revPembayaran;
11  }
12
13  // ---- getter setter
14  public ReversalPembayaran getRevPembayaran() { return revPembayaran
      ; }
15
16  public void setRevPembayaran(ReversalPembayaran revPembayaran) {
      this.revPembayaran = revPembayaran; }
17 }
```

20 *SOURCE CODE* StatusTrx.java

Isi dari *file* StatusTrx.java adalah kelas StatusTrx yang akan menampung informasi yang akan dikirimkan ke *client* terhadap *request* pencatatan pembayaran yang datang dari *client*. Kode dari kelas StatusTrx adalah sebagai berikut :

```
1 package lab.aikibo.model;
2
3 public class StatusTrx extends Status {
4
5   private PembayaranSppt byrSppt;
6
7   public StatusTrx() {}
8
9   public StatusTrx(int code, String message, PembayaranSppt byrSppt)
      {
10    super(code, message);
```

```
11      this.byrSppt = byrSppt;
12    }
13
14    // --- setter getter
15
16    public PembayaranSppt getByrSppt() {
17      return byrSppt;
18    }
19
20    public void setByrSppt(PembayaranSppt byrSppt) {
21      this.byrSppt = byrSppt;
22    }
23
24 }
```

21 *SOURCE CODE* PembayaranServices.java

Isi dari *file* PembayaranServices.java adalah *interface* PembayaranServices yang berfungsi sebagai objek umum dari layanan pencatatan pembayaran. Kode dari *interface* PembayaranServices ini adalah sebagai berikut :

```
1 package lab.aikibo.services;
2
3 import lab.aikibo.model.StatusTrx;
4 import java.math.BigInteger;
5
6 import org.joda.time.DateTime;
7
8 public interface PembayaranServices {
9   public StatusTrx prosesPembayaran(String nop, String thn, DateTime
      tglBayar, String ipClient);
10 }
```

22 *SOURCE CODE* PembayaranServices-Impl.java

Isi dari *file* PembayaranServicesImpl.java adalah kelas PembayaranServicesImpl yang merupakan implementasi dari *interface* PembayaranServices. Kelas ini nantinya akan melakukan pengolahan data apa saja yang perlu diambilkan dari basis data, dan seperti apa pengolahannya. Kode dari kelas PembayaranServices-Impl ini adalah sebagai berikut :

```
1  package lab.aikibo.services;
2
3  import org.springframework.beans.factory.annotation.Autowired;
4  import org.springframework.stereotype.Service;
5  import org.springframework.transaction.annotation.Transactional;
6
7  import lab.aikibo.dao.StoreProceduresDao;
8
9  import lab.aikibo.model.StatusTrx;
10
11 import org.joda.time.DateTime;
12
13 @Service("pembayaranServices")
14 @Transactional
15 public class PembayaranServicesImpl implements PembayaranServices {
16   @Autowired
17   private StoreProceduresDao spDao;
18
19   @Override
20   public StatusTrx prosesPembayaran(String nop, String thn, DateTime
       tglBayar, String ipClient) {
21     return spDao.prosesPembayaran(nop, thn, tglBayar.toDate(),
       ipClient);
```

```
22   }
23 }
```

23 *SOURCE CODE* ReversalServices.java

Isi dari *file* ReversalServices.java adalah *interface* ReversalServices yang merupakan bentuk umum dari pengolah data untuk *request reversal*. Kode dari *interface* ReversalServices ini adalah sebagai berikut :

```
1 package lab.aikibo.services;
2
3 import lab.aikibo.model.StatusRev;
4
5 public interface ReversalServices {
6   public StatusRev prosesReversal(String nop, String thn, String ntpd
      , String ipClient);
7 }
```

24 *SOURCE CODE* ReversalServicesImpl.java

Isi dari *file* ReversalServicesImpl.java adalah kelas ReversalServicesImpl yang merupakan implementasi dari *interface* kelas ReversalServices yang nantinya menangani pengolahan data terhadap *request reversal* dari *client*. Kode dari kelas ReversalServicesImpl ini adalah sebagai berikut :

```
1 package lab.aikibo.services;
2
3 import lab.aikibo.model.StatusRev;
4
5 import lab.aikibo.dao.StoreProceduresDao;
6
```

```
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.stereotype.Service;
9 import org.springframework.transaction.annotation.Transactional;
10
11 @Service("reversalServices")
12 @Transactional
13 public class ReversalServicesImpl implements ReversalServices {
14
15   @Autowired
16   private StoreProceduresDao spDao;
17
18   public StatusRev prosesReversal(String nop, String thn, String ntpd
       , String ipClient) {
19     return spDao.reversalPembayaran(nop, thn, ntpd, ipClient);
20   }
21 }
```

25 *SOURCE CODE* SpptServices.java

Isi dari *file* SpptServices.java adalah *interface* SpptService yang merupakan bentuk umum dari kelas yang menangani *request inquiry* data dari *client*. Kode dari *interface* SpptServices adalah sebagai berikut :

```
1 package lab.aikibo.services;
2
3 import lab.aikibo.model.StatusInq;
4
5 public interface SpptServices {
6
7   public StatusInq getSpptByNopThn(String nop, String thn, String
      ipClient);
8
```

```
9 }
```

26 *SOURCE CODE* SpptServicesImpl.java

Isi dari *file* SpptServicesImpl.java adalah kelas SpptServicesImpl yang merupakan implementasi dari *interface* SpptServices. Kode dari kelas SpptServicesImpl ini adalah sebagai berikut :

```
1 package lab.aikibo.services;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Service;
5 import org.springframework.transaction.annotation.Transactional;
6
7 import lab.aikibo.dao.SpptDao;
8 import lab.aikibo.dao.StoreProceduresDao;
9 import lab.aikibo.model.Sppt;
10 import lab.aikibo.model.SpptJ;
11 import lab.aikibo.model.StatusInq;
12 import lab.aikibo.controller.SpptRestController;
13
14 @Service("sppt")
15 @Transactional
16 public class SpptServicesImpl implements SpptServices {
17
18   @Autowired
19   private SpptDao spptDao;
20
21   @Autowired
22   private StoreProceduresDao spDao;
23
24   @Override
```

```
25    public StatusInq getSpptByNopThn(String nop, String thn, String
       ipClient) {
26      //return spptDao.inqSpptByNopThn(nop, thn);
27      return spDao.getDataSppt(nop, thn, ipClient);
28    }
29
30 }
```