LAPORAN PELAKSANAAN UJI COBA PROGRAM - WS PBB

6 Januari 2017

Priyanto Tamami, S.Kom.

1 PENDAHULUAN

Pengujian program atau aplikasi yang dilakukan berupa *unit testing* dan *integration testing* menggunakan *tools* yang sudah ada pada bahasa pemrograman Java yaitu JUnit.

JUnit ini nantinya akan melakukan testing per unit pada tiap fungsi / *method* yang membangun sistem aplikasi sehingga diharapkan tiap unit menghasilkan keluaran yang diharapkan.

2 OUTPUT PROGRAM

Karena menggunakan JUnit, maka keluarannya akan menghasilkan informasi `fail` dan `passed`.

Kondisi pengujian seperti dijelaskan sebelumnya bahwa akan dilakukan dengan *unit test* dan *integration test*, rinciannya adalah sebagai berikut :

A. *Unit Test*

Unit test yang dilakukan akan dibagi berdasarkan kelas/objek yang terbentuk, berikut adalah nama *file* yang melakukan *unit test*, nama *file* menunjukkan nama kelas/objek yang dilakukan *unit test*.

1. RootControllerTest

Pengujian kelas/objek `RootController` adalah untuk memastikan bahwa *request* yang masuk ke *server* mendapatkan respon yang diinginkan seperti nilai yang tertera pada basis data. Kode untuk melakukan pengujian pada kelas/objek `RootController` ini adalah sebagai berikut :

```
1  package lab.aikibo.controller;
2
3  import lab.aikibo.constant.StatusRespond;
4  import lab.aikibo.model.*;
5  import lab.aikibo.services.PembayaranServices;
6  import lab.aikibo.services.ReversalServices;
7  import lab.aikibo.services.SpptServices;
8  import org.joda.time.DateTime;
9  import org.junit.Before;
10 import org.junit.Test;
11 import org.junit.runner.RunWith;
12 import org.mockito.InjectMocks;
13 import org.mockito.Mock;
14 import org.mockito.Mockito;
15 import org.springframework.test.context.junit4.SpringRunner;
16
17 import javax.servlet.http.HttpServletRequest;
18 import java.math.BigInteger;
19
20 import static org.junit.Assert.assertEquals;
21 import static org.junit.Assert.assertNull;
22 import static org.mockito.Mockito.when;
23
24 /**
25  * Created by tamami.
26  */
27 @RunWith(SpringRunner.class)
28 public class RootControllerTest {
```

```
29
30      @InjectMocks
31      private RootController rootController = new RootController()
        ;
32
33      HttpServletRequest mock = Mockito.mock(HttpServletRequest.
        class);
34
35      @Mock
36      private SpptServices spptServices;
37      @Mock
38      private PembayaranServices byrServices;
39      @Mock
40      private ReversalServices revServices;
41
42      private StatusInq status;
43      private StatusInq statusInqGagalDataTidakAda;
44      private StatusInq statusInqError;
45      private Sppt sppt;
46
47      private StatusTrx statusTrxBerhasil;
48      private StatusTrx statusTrxNihil;
49      private StatusTrx statusTrxTerbayar;
50      private StatusTrx statusTrxBatal;
51      private StatusTrx statusTrxError;
52      private StatusTrx statusTrxhnPajakBukanAngka;
53      private StatusTrx statusTrxWaktuBayarLdWaktuCatat;
54      private PembayaranSppt byrSppt;
55
56      private StatusRev statusRevBerhasil;
57      private StatusRev statusRevNihil;
58      private StatusRev statusRevGanda;
59      private StatusRev statusRevError;
```

```
60      private ReversalPembayaran revSppt;
61
62      @Before
63      public void init() {
64          sppt = new Sppt("332901000100100010", "2013", "FULAN", "
        BREBES",
65                  new BigInteger("10000"), new BigInteger("0"));
66          status = new StatusInq(1, "Data Ditemukan", sppt);
67          statusInqGagalDataTidakAda = new StatusInq(StatusRespond
        .DATA_INQ_NIHIL, "Data Tidak Ditemukan", null);
68          statusInqError = new StatusInq(StatusRespond.
        DATABASE_ERROR, "Kesalahan DB", null);
69
70          byrSppt = new PembayaranSppt
        ("332901000100100010","2013","KODE_NTPD","4.1.1.12.1",
71                  new BigInteger("10000"), "4.1.1.12.2", new
        BigInteger("0"), "FULAN",
72                  "BREBES");
73          statusTrxBerhasil = new StatusTrx(StatusRespond.APPROVED
        , "Pembayaran Telah Tercatat", byrSppt);
74          statusTrxNihil = new StatusTrx(StatusRespond.
        TAGIHAN_TELAH_TERBAYAR,
75                  "Tagihan Telah Terbayar atau Pokok Pajak Nihil",
         null);
76          statusTrxTerbayar = new StatusTrx(StatusRespond.
        TAGIHAN_TELAH_TERBAYAR,
77                  "Tagihan Telah Terbayar", null);
78          statusTrxBatal = new StatusTrx(StatusRespond.
        JUMLAH_SETORAN_NIHIL,
79                  "Tagihan SPPT Telah Dibatalkan", null);
80          statusTrxError = new StatusTrx(StatusRespond.
        DATABASE_ERROR,
81                  "Kesalahan Server", null);
```

```
82
83          revSppt = new ReversalPembayaran
    ("332901000100100010","2013","KODE_NTPD");
84          statusRevBerhasil = new StatusRev(StatusRespond.APPROVED
    , "Proses Reversal Berhasil", revSppt);
85          statusRevNihil = new StatusRev(StatusRespond.
    DATA_INQ_NIHIL, "Data Yang Diminta Tidak Ada", null);
86          statusRevGanda = new StatusRev(StatusRespond.
    DATABASE_ERROR, "Data Tersebut Ganda", null);
87          statusRevError = new StatusRev(StatusRespond.
    DATABASE_ERROR, "Kesalahan Server", null);
88      }
89
90      @Test
91      public void testInquirySpptBerhasil() {
92          when(spptServices.getSpptByNopThn
    ("332901000100100010","2013",null))
93                  .thenReturn(status);
94
95          assertEquals(1, rootController.getSppt
    ("332901000100100010","2013", mock).getCode());
96          assertEquals("Data Ditemukan",
97                  rootController.getSppt("332901000100100010",
    "2013", mock).getMessage());
98          assertEquals("332901000100100010",
99                  rootController.getSppt
    ("332901000100100010","2013", mock).getSppt().getNop());
100         assertEquals("2013",
101                 rootController.getSppt
    ("332901000100100010","2013", mock).getSppt().getThn());
102         assertEquals("FULAN",
103                 rootController.getSppt
    ("332901000100100010","2013", mock).getSppt().getNama());
```

```
104         assertEquals("BREBES",
105                 rootController.getSppt
    ("332901000100100010","2013", mock).getSppt().getAlamatOp())
    ;
106         assertEquals(new BigInteger("10000"),
107                 rootController.getSppt("332901000100100010",
    "2013", mock).getSppt().getPokok());
108         assertEquals(new BigInteger("0"),
109                 rootController.getSppt("332901000100100010",
    "2013", mock).getSppt().getDenda());
110     }
111
112     @Test
113     public void testInquirySpptGagalKarnaNihil() {
114         when(spptServices.getSpptByNopThn
    ("332901000100100020","2013",null))
115                 .thenReturn(statusInqGagalDataTidakAda);
116
117         assertEquals(10,
118                 rootController.getSppt
    ("332901000100100020","2013", mock).getCode());
119         assertEquals("Data Tidak Ditemukan",
120                 rootController.getSppt
    ("332901000100100020","2013", mock).getMessage());
121         assertNull(rootController.getSppt
    ("332901000100100020","2013", mock).getSppt());
122     }
123
124     @Test
125     public void testInquiryDbError() {
126         when(spptServices.getSpptByNopThn
    ("332901000100100030","2013",null))
127                 .thenReturn(statusInqError);
```

```
128
129         assertEquals(4,
130                 rootController.getSppt
    ("332901000100100030","2013", mock).getCode());
131         assertEquals("Kesalahan DB",
132                 rootController.getSppt
    ("332901000100100030","2013", mock).getMessage());
133         assertNull(rootController.getSppt
    ("332901000100100030","2013", mock).getSppt());
134     }
135
136     @Test
137     public void testInquiryThnPajakBukanAngka() {
138         assertEquals(36,
139                 rootController.getSppt("332901000100100010","2
    a13", mock).getCode());
140         assertEquals("Tahun Pajak Mengandung Karakter Bukan
    Angka",
141                 rootController.getSppt("332901000100100010","2
    a13", mock).getMessage());
142         assertNull(rootController.getSppt
    ("332901000100100010","2a13", mock).getSppt());
143     }
144
145     @Test
146     public void testTrxWaktuBayarLdWaktuCatat() {
147         assertEquals(StatusRespond.
    TGL_JAM_BAYAR_LD_TGL_JAM_KIRIM,
148                 rootController.prosesPembayaran
    ("332901000100100010","2013","22122017",
149                         "1000", mock).getCode());
150         assertEquals("Tanggal atau jam pada saat dibayarkan
    melebihi tanggal dan jam saat ini",
```

```
151                 rootController.prosesPembayaran
    ("332901000100100010","2013","22122017",
152                         "1000", mock).getMessage());
153         assertNull(rootController.prosesPembayaran
    ("332901000100100010","2013","22122017",
154                 "1000", mock).getByrSppt());
155     }
156
157     @Test
158     public void testTrxSpptSukses() {
159         when(byrServices.prosesPembayaran
    ("332901000100100010","2013",
160                 new DateTime(2016, 12, 19, 10, 0),null))
161                 .thenReturn(statusTrxBerhasil);
162
163         assertEquals(1,
164                 rootController.prosesPembayaran
    ("332901000100100010","2013","19122016",
165                         "1000", mock).getCode());
166         assertEquals("Pembayaran Telah Tercatat",
167                 rootController.prosesPembayaran
    ("332901000100100010","2013","19122016",
168                         "1000", mock).getMessage());
169         assertEquals("332901000100100010",
170                 rootController.prosesPembayaran
    ("332901000100100010","2013","19122016",
171                         "1000", mock).getByrSppt().getNop());
172         assertEquals("2013",
173                 rootController.prosesPembayaran
    ("332901000100100010","2013","19122016",
174                         "1000", mock).getByrSppt().getThn());
175         assertEquals("KODE_NTPD",
176                 rootController.prosesPembayaran
```

```
    ("332901000100100010","2013","19122016",
177                         "1000", mock).getByrSppt().getNtpd());
178         assertEquals("4.1.1.12.1",
179                 rootController.prosesPembayaran
    ("332901000100100010","2013","19122016",
180                         "1000", mock).getByrSppt().
    getMataAnggaranPokok());
181         assertEquals(new BigInteger("10000"),
182                 rootController.prosesPembayaran
    ("332901000100100010","2013","19122016",
183                         "1000", mock).getByrSppt().getPokok());
184         assertEquals("4.1.1.12.2",
185                 rootController.prosesPembayaran
    ("332901000100100010","2013","19122016",
186                         "1000", mock).getByrSppt().
    getMataAnggaranSanksi());
187         assertEquals(new BigInteger("0"),
188                 rootController.prosesPembayaran
    ("332901000100100010", "2013","19122016",
189                         "1000", mock).getByrSppt().getSanksi());
190         assertEquals("FULAN",
191                 rootController.prosesPembayaran
    ("332901000100100010", "2013", "19122016",
192                         "1000", mock).getByrSppt().getNamaWp());
193         assertEquals("BREBES",
194                 rootController.prosesPembayaran
    ("332901000100100010","2013","19122016",
195                         "1000", mock).getByrSppt().getAlamatOp()
    );
196     }
197
198
199
```

```
200     @Test
201     public void testTrxNihil() {
202         when(byrServices.prosesPembayaran
        ("332901000100100010","2013",
203                 new DateTime(2016, 12, 19, 10, 0),null))
204                 .thenReturn(statusTrxNihil);
205
206         assertEquals(StatusRespond.TAGIHAN_TELAH_TERBAYAR,
207                 rootController.prosesPembayaran
        ("332901000100100010","2013",
208                         "19122016","1000", mock).getCode());
209         assertEquals("Tagihan Telah Terbayar atau Pokok Pajak
        Nihil",
210                 rootController.prosesPembayaran
        ("332901000100100010","2013","19122016",
211                         "1000", mock).getMessage());
212         assertNull(rootController.prosesPembayaran
        ("332901000100100010","2013","19122016",
213                 "1000", mock).getByrSppt());
214     }
215
216     @Test
217     public void testTrxTerbayar() {
218         when(byrServices.prosesPembayaran
        ("332901000100100010","2013",
219                 new DateTime(2016,12,19,10,0), null)).thenReturn
        (statusTrxTerbayar);
220
221         assertEquals(StatusRespond.TAGIHAN_TELAH_TERBAYAR,
222                 rootController.prosesPembayaran
        ("332901000100100010","2013","19122016",
223                         "1000", mock).getCode());
224         assertEquals("Tagihan Telah Terbayar",
```

```
225                 rootController.prosesPembayaran
    ("332901000100100010","2013","19122016",
226                         "1000", mock).getMessage());
227         assertNull(rootController.prosesPembayaran
    ("332901000100100010","2013","19122016",
228                 "1000", mock).getByrSppt());
229     }
230
231     @Test
232     public void testTrxBatal() {
233         when(byrServices.prosesPembayaran
    ("332901000100100010","2013",
234                 new DateTime(2016,12,19,10,0), null)).thenReturn
    (statusTrxBatal);
235
236         assertEquals(StatusRespond.JUMLAH_SETORAN_NIHIL,
237                 rootController.prosesPembayaran
    ("332901000100100010","2013","19122016",
238                         "1000", mock).getCode());
239         assertEquals("Tagihan SPPT Telah Dibatalkan",
240                 rootController.prosesPembayaran
    ("332901000100100010","2013","19122016",
241                         "1000", mock).getMessage());
242         assertNull(rootController.prosesPembayaran
    ("332901000100100010","2013","19122016",
243                 "1000", mock).getByrSppt());
244     }
245
246     @Test
247     public void testTrxError() {
248         when(byrServices.prosesPembayaran
    ("332901000100100010","2013",
249                 new DateTime(2016,12,19,10,0), null)).thenReturn
```

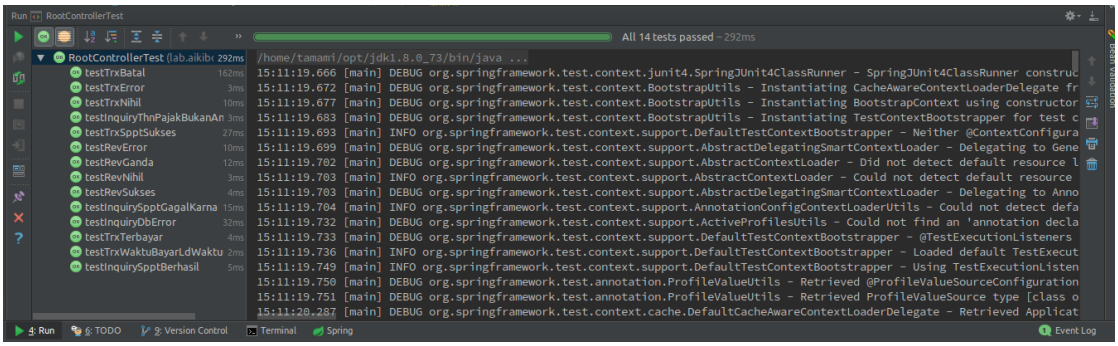
```
    (statusTrxError);
250
251         assertEquals(StatusRespond.DATABASE_ERROR,
252                 rootController.prosesPembayaran
    ("332901000100100010","2013","19122016",
253                         "1000", mock).getCode());
254         assertEquals("Kesalahan Server",
255                 rootController.prosesPembayaran
    ("332901000100100010", "2013","19122016",
256                         "1000", mock).getMessage());
257         assertNull(rootController.prosesPembayaran
    ("332901000100100010","2013","19122016",
258                 "1000", mock).getByrSppt());
259     }
260
261     @Test
262     public void testRevSukses() {
263         when(revServices.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", null))
264                 .thenReturn(statusRevBerhasil);
265
266         assertEquals(StatusRespond.APPROVED,
267                 rootController.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", mock).getCode());
268         assertEquals("Proses Reversal Berhasil",
269                 rootController.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", mock).getMessage()
    );
270         assertEquals("332901000100100010",
271                 rootController.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", mock)
272                         .getRevPembayaran().getNop());
273         assertEquals("2013",
```

```
274                 rootController.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", mock)
275                         .getRevPembayaran().getThn());
276         assertEquals("KODE_NTPD",
277                 rootController.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", mock)
278                         .getRevPembayaran().getNtpd());
279     }
280
281     @Test
282     public void testRevNihil() {
283         when(revServices.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", null))
284                 .thenReturn(statusRevNihil);
285
286         assertEquals(StatusRespond.DATA_INQ_NIHIL,
287                 rootController.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", mock).getCode());
288         assertEquals("Data Yang Diminta Tidak Ada",
289                 rootController.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", mock).getMessage()
    );
290         assertNull(rootController.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", mock).
291                 getRevPembayaran());
292     }
293
294     @Test
295     public void testRevGanda() {
296         when(revServices.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", null))
297                 .thenReturn(statusRevGanda);
298
```

```
299         assertEquals(StatusRespond.DATABASE_ERROR,
300                 rootController.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", mock).getCode());
301         assertEquals("Data Tersebut Ganda",
302                 rootController.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", mock).getMessage()
    );
303         assertNull(rootController.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", mock)
304                 .getRevPembayaran());
305     }
306
307     @Test
308     public void testRevError() {
309         when(revServices.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", null))
310                 .thenReturn(statusRevError);
311
312         assertEquals(StatusRespond.DATABASE_ERROR,
313                 rootController.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", mock).getCode());
314         assertEquals("Kesalahan Server",
315                 rootController.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", mock).getMessage()
    );
316         assertNull(rootController.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", mock)
317                 .getRevPembayaran());
318     }
319
320 }
321
```

Agar tidak mengganggu kondisi data pada sistem basis data yang berjalan, maka *unit test* ini menggunakan data model sebagaimana disiapkan pada *method* `init()`, kemudian datanya akan di *mock* ke dalam *service* yang menangani, seperti misalkan pada kasus *inquiry* data SPPT, maka data model akan di *mock* ke dalam objek `SpptServices`, yang apabila diujikan ada *request* masuk melalui *method* `getSppt`, maka seharusnya sistem akan menghasilkan data persis seperti apa yang telah di *mock* ke dalam *service*.

Hasil dari *unit test* tersebut seperti terlihat pada gambar 1



Gambar 1: Hasil *Unit Test* Untuk Kelas `RootController`

Dari hasil tersebut menunjukkan bahwa pengujian telah berhasil dilakukan.

2. StoreProceduresDaoImplTest

   Pengujian pada kelas `StoreProceduresDaoImplTest` dilakukan untuk memastikan bahwa proses di dalam kelas `StoreProceduresDaoImpl` berjalan sebagaimana seharusnya.

   Kode pengujian untuk kelas `StoreProceduresDaoImpl` ini adalah sebagai berikut :

```
1 package lab.aikibo.dao;
2
3 import lab.aikibo.constant.StatusRespond;
4 import lab.aikibo.model.*;
```

```
5  import org.joda.time.DateTime;
6  import org.junit.Before;
7  import org.junit.Test;
8  import org.mockito.Mock;
9  import org.mockito.Mockito;
10
11 import java.math.BigInteger;
12
13 import static org.junit.Assert.assertEquals;
14 import static org.junit.Assert.assertNull;
15 import static org.mockito.Mockito.when;
16
17 /**
18  * Created by tamami.
19  */
20 public class StoreProceduresDaoImplTest {
21
22     @Mock
23     private StoreProceduresDao spDao;
24
25     private StatusInq statusInq;
26     private StatusInq statusInqNihil;
27     private StatusInq statusInqError;
28     private Sppt sppt;
29
30     private StatusTrx statusTrxBerhasil;
31     private StatusTrx statusTrxNihil;
32     private StatusTrx statusTrxTerbayar;
33     private StatusTrx statusTrxBatal;
34     private StatusTrx statusTrxError;
35     private PembayaranSppt byrSppt;
36
37     private StatusRev statusRevBerhasil;
```

```
38      private StatusRev statusRevNihil;
39      private StatusRev statusRevGanda;
40      private StatusRev statusRevError;
41      private ReversalPembayaran revSppt;
42
43
44      @Before
45      public void init() {
46          spDao = Mockito.mock(StoreProceduresDaoImpl.class);
47
48          sppt = new Sppt("332901000100100010", "2013", "FULAN", "
    BREBES",
49                  new BigInteger("10000"), new BigInteger("0"));
50          statusInq = new StatusInq(StatusRespond.APPROVED, "Data
    Ditemukan", sppt);
51          statusInqNihil = new StatusInq(StatusRespond.
    DATA_INQ_NIHIL, "Data Tidak Ditemukan", null);
52          statusInqError = new StatusInq(StatusRespond.
    DATABASE_ERROR, "Kesalahan DB", null);
53
54          byrSppt = new PembayaranSppt
    ("332901000100100010","2013","KODE_NTPD","4.1.1.12.1",
55                  new BigInteger("10000"), "4.1.1.12.2", new
    BigInteger("0"), "FULAN",
56                  "BREBES");
57          statusTrxBerhasil = new StatusTrx(StatusRespond.APPROVED
    , "Transaksi Telah Tercatat", byrSppt);
58          statusTrxNihil = new StatusTrx(StatusRespond.
    JUMLAH_SETORAN_NIHIL, "Data Yang Diminta Tidak Ada",
59                  null);
60          statusTrxTerbayar = new StatusTrx(StatusRespond.
    TAGIHAN_TELAH_TERBAYAR, "Tagihan Telah Terbayar", null);
61          statusTrxBatal = new StatusTrx(StatusRespond.
```

```
    JUMLAH_SETORAN_NIHIL, "Tagihan Telah Dibatalkan", null);
62          statusTrxError = new StatusTrx(StatusRespond.
    DATABASE_ERROR, "Kesalahan Server", null);
63
64          revSppt = new ReversalPembayaran
    ("332901000100100010","2013","KODE_NTPD");
65          statusRevBerhasil = new StatusRev(StatusRespond.APPROVED
    , "Reversal Telah Berhasil Dilakukan", revSppt);
66          statusRevNihil = new StatusRev(StatusRespond.
    DATA_INQ_NIHIL, "Data Yang Diminta Tidak Ada",
67                  null);
68          statusRevGanda = new StatusRev(StatusRespond.
    DATABASE_ERROR, "Data Transaksi Tercatat Ganda",
69                  null);
70          statusRevError = new StatusRev(StatusRespond.
    DATABASE_ERROR, "Kesalahan Server",
71                  null);
72      }
73
74      @Test
75      public void testInqBerhasil() {
76          when(spDao.getDataSppt("332901000100100010","2013", null
    ))
77                  .thenReturn(statusInq);
78
79          assertEquals(StatusRespond.APPROVED,
80                  spDao.getDataSppt("332901000100100010","2013",
    null).getCode());
81          assertEquals("Data Ditemukan",
82                  spDao.getDataSppt("332901000100100010","2013",
    null).getMessage());
83          assertEquals("332901000100100010",
84                  spDao.getDataSppt("332901000100100010","2013",
```

```
    null).getSppt().getNop());
85          assertEquals("2013",
86                  spDao.getDataSppt("332901000100100010","2013",
    null).getSppt().getThn());
87          assertEquals("FULAN",
88                  spDao.getDataSppt("332901000100100010","2013",
    null).getSppt().getNama());
89          assertEquals("BREBES",
90                  spDao.getDataSppt("332901000100100010","2013",
    null).getSppt().getAlamatOp());
91          assertEquals(new BigInteger("10000"),
92                  spDao.getDataSppt("332901000100100010","2013",
    null).getSppt().getPokok());
93          assertEquals(new BigInteger("0"),
94                  spDao.getDataSppt("332901000100100010","2013",
    null).getSppt().getDenda());
95      }
96
97      @Test
98      public void testInqNihil() {
99          when(spDao.getDataSppt("332901000100100010","2013",null)
    )
100                 .thenReturn(statusInqNihil);
101
102         assertEquals(StatusRespond.DATA_INQ_NIHIL,
103                 spDao.getDataSppt("332901000100100010","2013",
    null).getCode());
104         assertEquals("Data Tidak Ditemukan",
105                 spDao.getDataSppt("332901000100100010","2013",
    null).getMessage());
106         assertNull(spDao.getDataSppt
    ("332901000100100010","2013", null).getSppt());
107     }
```

```
108
109     @Test
110     public void testInqError() {
111         when(spDao.getDataSppt("332901000100100010", "2013",
    null))
112                 .thenReturn(statusInqError);
113
114         assertEquals(StatusRespond.DATABASE_ERROR,
115                 spDao.getDataSppt("332901000100100010","2013",
    null).getCode());
116         assertEquals("Kesalahan DB",
117                 spDao.getDataSppt("332901000100100010","2013",
    null).getMessage());
118         assertNull(spDao.getDataSppt
    ("332901000100100010","2013",null).getSppt());
119     }
120
121     @Test
122     public void testTrxSukses() {
123         when(spDao.prosesPembayaran("332901000100100010","2013",
124                 new DateTime(2016,12,19,10,0).toDate(), null))
125                 .thenReturn(statusTrxBerhasil);
126
127         assertEquals(StatusRespond.APPROVED,
128                 spDao.prosesPembayaran
    ("332901000100100010","2013",
129                         new DateTime(2016,12,19,10,0).toDate(),
    null).getCode());
130         assertEquals("Transaksi Telah Tercatat",
131                 spDao.prosesPembayaran
    ("332901000100100010","2013",
132                         new DateTime(2016,12,19,10,0).toDate(),
    null).getMessage());
```

```
133         assertEquals("332901000100100010",
134                 spDao.prosesPembayaran
    ("332901000100100010","2013",
135                         new DateTime(2016,12,19,10,0).toDate(),
    null).getByrSppt().getNop());
136         assertEquals("2013",
137                 spDao.prosesPembayaran
    ("332901000100100010","2013",
138                         new DateTime(2016,12,19,10,0).toDate(),
    null).getByrSppt().getThn());
139         assertEquals("KODE_NTPD",
140                 spDao.prosesPembayaran
    ("332901000100100010","2013",
141                         new DateTime(2016,12,19,10,0).toDate(),
    null).getByrSppt().getNtpd());
142         assertEquals("4.1.1.12.1",
143                 spDao.prosesPembayaran
    ("332901000100100010","2013",
144                         new DateTime(2016,12,19,10,0).toDate(),
    null).getByrSppt()
145                         .getMataAnggaranPokok());
146         assertEquals(new BigInteger("10000"),
147                 spDao.prosesPembayaran
    ("332901000100100010","2013",
148                         new DateTime(2016,12,19,10,0).toDate(),
    null).getByrSppt()
149                         .getPokok());
150         assertEquals("4.1.1.12.2",
151                 spDao.prosesPembayaran
    ("332901000100100010","2013",
152                         new DateTime(2016,12,19,10,0).toDate(),
    null).getByrSppt()
153                         .getMataAnggaranSanksi());
```

```
154         assertEquals(new BigInteger("0"),
155                 spDao.prosesPembayaran
    ("332901000100100010","2013",
156                         new DateTime(2016,12,19,10,0).toDate(),
    null).getByrSppt().getSanksi());
157         assertEquals("FULAN",
158                 spDao.prosesPembayaran
    ("332901000100100010","2013",
159                         new DateTime(2016,12,19,10,0).toDate(),
    null).getByrSppt().getNamaWp());
160         assertEquals("BREBES",
161                 spDao.prosesPembayaran
    ("332901000100100010","2013",
162                         new DateTime(2016,12,19,10,0).toDate(),
    null).getByrSppt().getAlamatOp());
163     }
164
165     @Test
166     public void testTrxNihil() {
167         when(spDao.prosesPembayaran("332901000100100010","2013",
168                 new DateTime(2016,12,19,10,0).toDate(), null))
169                 .thenReturn(statusTrxNihil);
170
171         assertEquals(StatusRespond.JUMLAH_SETORAN_NIHIL,
172                 spDao.prosesPembayaran
    ("332901000100100010","2013",
173                         new DateTime(2016,12,19,10,0).toDate(),
    null).getCode());
174         assertEquals("Data Yang Diminta Tidak Ada",
175                 spDao.prosesPembayaran
    ("332901000100100010","2013",
176                         new DateTime(2016,12,19,10,0).toDate(),
    null).getMessage());
```

```
177         assertNull(spDao.prosesPembayaran
    ("332901000100100010","2013",
178                 new DateTime(2016,12,19,10,0).toDate(), null).
    getByrSppt());
179     }
180
181     @Test
182     public void testTrxTerbayar() {
183         when(spDao.prosesPembayaran("332901000100100010","2013",
184                 new DateTime(2016,12,19,10,0).toDate(), null))
185                 .thenReturn(statusTrxTerbayar);
186
187         assertEquals(StatusRespond.TAGIHAN_TELAH_TERBAYAR,
188                 spDao.prosesPembayaran
    ("332901000100100010","2013",
189                         new DateTime(2016,12,19,10,0).toDate(),
    null).getCode());
190         assertEquals("Tagihan Telah Terbayar",
191                 spDao.prosesPembayaran
    ("332901000100100010","2013",
192                         new DateTime(2016,12,19,10,0).toDate(),
    null).getMessage());
193         assertNull(spDao.prosesPembayaran
    ("332901000100100010","2013",
194                         new DateTime(2016,12,19,10,0).toDate(),
    null).getByrSppt());
195     }
196
197     @Test
198     public void testTrxBatal() {
199         when(spDao.prosesPembayaran("332901000100100010","2013",
200                 new DateTime(2016,12,19,10,0).toDate(), null))
201                 .thenReturn(statusTrxBatal);
```

```
202
203         assertEquals(StatusRespond.JUMLAH_SETORAN_NIHIL,
204                 spDao.prosesPembayaran
    ("332901000100100010","2013",
205                         new DateTime(2016,12,19,10,00).toDate(),
    null).getCode());
206         assertEquals("Tagihan Telah Dibatalkan",
207                 spDao.prosesPembayaran
    ("332901000100100010","2013",
208                         new DateTime(2016,12,19,10,0).toDate(),
    null).getMessage());
209         assertNull(spDao.prosesPembayaran
    ("332901000100100010","2013",
210                         new DateTime(2016,12,19,10,0).toDate(),
    null).getByrSppt());
211     }
212
213     @Test
214     public void testTrxError() {
215         when(spDao.prosesPembayaran("332901000100100010","2013",
216                 new DateTime(2016,12,19,10,0).toDate(), null))
217                 .thenReturn(statusTrxError);
218
219         assertEquals(StatusRespond.DATABASE_ERROR,
220                 spDao.prosesPembayaran
    ("332901000100100010","2013",
221                         new DateTime(2016,12,19,10,0).toDate(),
    null).getCode());
222         assertEquals("Kesalahan Server",
223                 spDao.prosesPembayaran
    ("332901000100100010","2013",
224                         new DateTime(2016,12,19,10,0).toDate(),
    null).getMessage());
```

```
225         assertNull(spDao.prosesPembayaran
    ("332901000100100010","2013",
226                 new DateTime(2016,12,19,10,0).toDate(), null).
    getByrSppt());
227     }
228
229     @Test
230     public void testRevSukses() {
231         when(spDao.reversalPembayaran
    ("332901000100100010","2013","KODE_NTPD", null))
232                 .thenReturn(statusRevBerhasil);
233
234         assertEquals(StatusRespond.APPROVED,
235                 spDao.reversalPembayaran
    ("332901000100100010","2013","KODE_NTPD", null).getCode());
236         assertEquals("Reversal Telah Berhasil Dilakukan",
237                 spDao.reversalPembayaran("332901000100100010",
    "2013", "KODE_NTPD", null).getMessage());
238         assertEquals("332901000100100010",
239                 spDao.reversalPembayaran
    ("332901000100100010","2013", "KODE_NTPD", null)
240             .getRevPembayaran().getNop());
241         assertEquals("2013",
242                 spDao.reversalPembayaran
    ("332901000100100010","2013","KODE_NTPD", null)
243                         .getRevPembayaran().getThn());
244         assertEquals("KODE_NTPD",
245                 spDao.reversalPembayaran
    ("332901000100100010","2013", "KODE_NTPD", null)
246                         .getRevPembayaran().getNtpd());
247
248     }
249
```

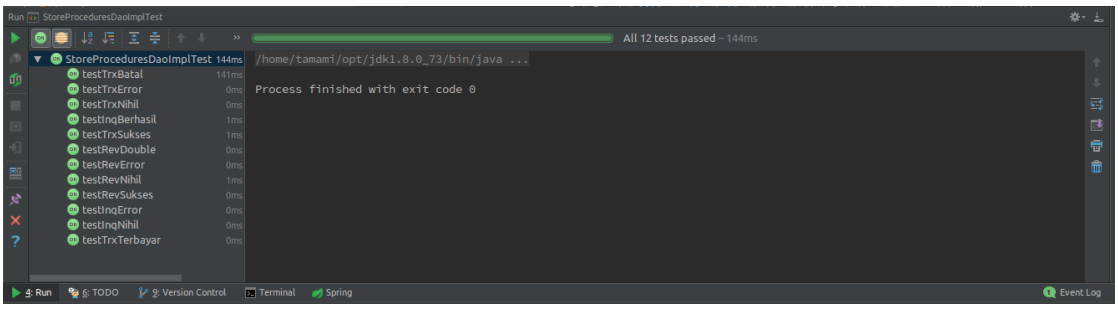
```
250     @Test
251     public void testRevNihil() {
252         when(spDao.reversalPembayaran
        ("332901000100100010","2013","KODE_NTPD", null))
253                 .thenReturn(statusRevNihil);
254
255         assertEquals(StatusRespond.DATA_INQ_NIHIL,
256                 spDao.reversalPembayaran
        ("332901000100100010","2013","KODE_NTPD", null)
257                         .getCode());
258         assertEquals("Data Yang Diminta Tidak Ada",
259                 spDao.reversalPembayaran
        ("332901000100100010","2013","KODE_NTPD", null)
260                         .getMessage());
261         assertNull(spDao.reversalPembayaran
        ("332901000100100010","2013","KODE_NTPD", null)
262                 .getRevPembayaran());
263     }
264
265     @Test
266     public void testRevDouble() {
267         when(spDao.reversalPembayaran
        ("332901000100100010","2013","KODE_NTPD", null))
268                 .thenReturn(statusRevGanda);
269
270         assertEquals(StatusRespond.DATABASE_ERROR,
271                 spDao.reversalPembayaran
        ("332901000100100010","2013","KODE_NTPD", null)
272                         .getCode());
273         assertEquals("Data Transaksi Tercatat Ganda",
274                 spDao.reversalPembayaran
        ("332901000100100010","2013","KODE_NTPD", null)
275                         .getMessage());
```

```
276             assertNull(spDao.reversalPembayaran
        ("332901000100100010","2013","KODE_NTPD", null)
277                     .getRevPembayaran());
278         }
279
280         @Test
281         public void testRevError() {
282             when(spDao.reversalPembayaran
        ("332901000100100010","2013","KODE_NTPD", null))
283                     .thenReturn(statusRevError);
284
285             assertEquals(StatusRespond.DATABASE_ERROR,
286                     spDao.reversalPembayaran
        ("332901000100100010","2013","KODE_NTPD", null)
287                             .getCode());
288             assertEquals("Kesalahan Server",
289                     spDao.reversalPembayaran
        ("332901000100100010","2013","KODE_NTPD",null)
290                             .getMessage());
291             assertNull(spDao.reversalPembayaran
        ("332901000100100010","2013","KODE_NTPD",null)
292                     .getRevPembayaran());
293         }
294
295 }
296
```

Karena kelas `StoreProceduresDao` berkomunikasi langsung dengan sistem basis data, tugas *unit test* ini adalah melakukan pengujian bahwa proses yang dihasilkan sesuai dengan isi basis data tanpa menyentuh atau berhubungan dengan sistem basis data, oleh karena itu pengujian pada kelas/objek `StoreProceduresDaoImplTest` menggunakan *mocking* untuk

membuat data model.

Dari hasil pengujian tersebut, didapatkan hasil seperti terlihat pada gambar 2



Gambar 2: Hasil *Unit Test* Pada Kelas `StoreProceduresDaoImpl`

Dari gambar tersebut menunjukkan bahwa hasil pengujian yang dilakukan berhasil sesuai dengan apa yang diharapkan.

3. PembayaranServicesImplTest

   Kelas/objek `PembayaranServicesImplTest` digunakan untuk melakukan pengujian pada kelas `PembayaranServicesImpl` yang menghubungkan bagian *view* dan *model* pada konsep MVC. Perlu dipastikan bahwa hasil dari pengambilan data pada sistem basis data diterima dengan benar oleh kelas/objek `RootController`, berikut adalah kode untuk melakukan *unit testing* pada kelas `PembayaranServicesImplTest` :

```
1 package lab.aikibo.services;
2
3 import lab.aikibo.constant.StatusRespond;
4 import lab.aikibo.dao.StoreProceduresDaoImpl;
5 import lab.aikibo.model.PembayaranSppt;
6 import lab.aikibo.model.StatusTrx;
7 import org.joda.time.DateTime;
8 import org.junit.Before;
```

```
9  import org.junit.Test;
10 import org.junit.runner.RunWith;
11 import org.mockito.InjectMocks;
12 import org.mockito.Mock;
13 import org.springframework.test.context.junit4.SpringRunner;
14
15 import java.math.BigInteger;
16
17 import static org.junit.Assert.assertEquals;
18 import static org.junit.Assert.assertNull;
19 import static org.mockito.Mockito.when;
20
21 /**
22  * Created by tamami.
23  */
24 @RunWith(SpringRunner.class)
25 public class PembayaranServicesImplTest {
26
27     @InjectMocks
28     private PembayaranServicesImpl byrServices = new
       PembayaranServicesImpl();
29
30     @Mock
31     private StoreProceduresDaoImpl spDao;
32
33     private StatusTrx statusTrxBerhasil;
34     private StatusTrx statusTrxNihil;
35     private StatusTrx statusTrxTerbayar;
36     private StatusTrx statusTrxBatal;
37     private StatusTrx statusTrxError;
38     private StatusTrx statusTrxhnPajakBukanAngka;
39     private StatusTrx statusTrxWaktuBayarLdWaktuCatat;
40     private PembayaranSppt byrSppt;
```

```
41
42      @Before
43      public void init() {
44          byrSppt = new PembayaranSppt
        ("332901000100100010","2013","KODE_NTPD","4.1.1.12.1",
45                  new BigInteger("10000"), "4.1.1.12.2", new
        BigInteger("0"), "FULAN",
46                  "BREBES");
47          statusTrxBerhasil = new StatusTrx(StatusRespond.APPROVED
        , "Transaksi Telah Tercatat", byrSppt);
48          statusTrxNihil = new StatusTrx(StatusRespond.
        JUMLAH_SETORAN_NIHIL, "Data Yang Diminta Tidak Ada",
49                  null);
50          statusTrxTerbayar = new StatusTrx(StatusRespond.
        TAGIHAN_TELAH_TERBAYAR, "Tagihan Telah Terbayar", null);
51          statusTrxBatal = new StatusTrx(StatusRespond.
        JUMLAH_SETORAN_NIHIL, "Tagihan Telah Dibatalkan", null);
52          statusTrxError = new StatusTrx(StatusRespond.
        DATABASE_ERROR, "Kesalahan Server", null);
53      }
54
55      @Test
56      public void testTrxBerhasil() {
57          when(spDao.prosesPembayaran("332901000100100010","2013",
58                  new DateTime(2016,12,20,10,0).toDate(), null)).
        thenReturn(statusTrxBerhasil);
59
60          assertEquals(StatusRespond.APPROVED,
61                  byrServices.prosesPembayaran
        ("332901000100100010","2013",
62                          new DateTime(2016,12,20,10,0), null).
        getCode());
63          assertEquals("Transaksi Telah Tercatat",
```

```
64                  byrServices.prosesPembayaran
    ("332901000100100010","2013",
65                          new DateTime(2016,12,20,10,0), null).
    getMessage());
66          assertEquals("332901000100100010",
67                  byrServices.prosesPembayaran
    ("332901000100100010","2013",
68                          new DateTime(2016,12,20,10,0), null).
    getByrSppt().getNop());
69          assertEquals("2013",
70                  byrServices.prosesPembayaran
    ("332901000100100010","2013",
71                          new DateTime(2016,12,20,10,0), null).
    getByrSppt().getThn());
72          assertEquals("KODE_NTPD",
73                  byrServices.prosesPembayaran
    ("332901000100100010","2013",
74                          new DateTime(2016,12,20,10,0), null).
    getByrSppt().getNtpd());
75          assertEquals("4.1.1.12.1",
76                  byrServices.prosesPembayaran
    ("332901000100100010","2013",
77                          new DateTime(2016,12,20,10,0),null).
    getByrSppt().getMataAnggaranPokok());
78          assertEquals(new BigInteger("10000"),
79                  byrServices.prosesPembayaran
    ("332901000100100010","2013",
80                          new DateTime(2016,12,20,10,0),null).
    getByrSppt().getPokok());
81          assertEquals("4.1.1.12.2",
82                  byrServices.prosesPembayaran
    ("332901000100100010","2013",
83                          new DateTime(2016,12,20,10,0), null).
```

```
    getByrSppt().getMataAnggaranSanksi());
84          assertEquals(new BigInteger("0"),
85                  byrServices.prosesPembayaran
    ("332901000100100010","2013",
86                          new DateTime(2016,12,20,10,0), null).
    getByrSppt().getSanksi());
87          assertEquals("FULAN",
88                  byrServices.prosesPembayaran
    ("332901000100100010","2013",
89                          new DateTime(2016,12,20,10,0),null).
    getByrSppt().getNamaWp());
90          assertEquals("BREBES",
91                  byrServices.prosesPembayaran
    ("332901000100100010","2013",
92                          new DateTime(2016,12,20,10,0),null).
    getByrSppt().getAlamatOp());
93
94      }
95
96      @Test
97      public void testTrxNihil() {
98          when(spDao.prosesPembayaran("332901000100100010","2013",
99                  new DateTime(2016,12,20,10,0).toDate(), null)).
    thenReturn(statusTrxNihil);
100
101         assertEquals(StatusRespond.JUMLAH_SETORAN_NIHIL,
102                 byrServices.prosesPembayaran
    ("332901000100100010","2013",
103                         new DateTime(2016,12,20,10,0), null).
    getCode());
104         assertEquals("Data Yang Diminta Tidak Ada",
105                 byrServices.prosesPembayaran
    ("332901000100100010","2013",
```

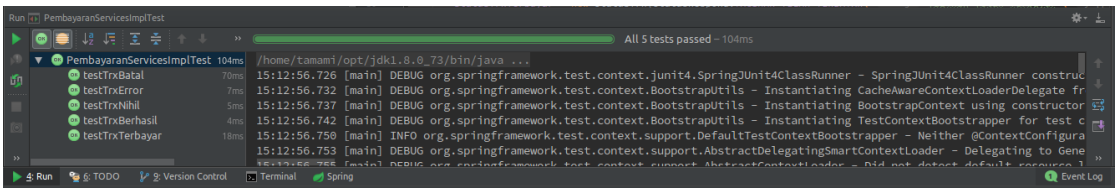
```
106                         new DateTime(2016,12,20,10,0), null).
    getMessage());
107         assertNull(byrServices.prosesPembayaran
    ("332901000100100010","2013",
108                 new DateTime(2016,12,20,10,0),null).getByrSppt()
    );
109     }
110
111     @Test
112     public void testTrxTerbayar() {
113         when(spDao.prosesPembayaran("332901000100100010","2013",
114                 new DateTime(2016,12,20,10,0).toDate(), null)).
    thenReturn(statusTrxTerbayar);
115
116         assertEquals(StatusRespond.TAGIHAN_TELAH_TERBAYAR,
117                 byrServices.prosesPembayaran
    ("332901000100100010","2013",
118                         new DateTime(2016,12,20,10,0),null).
    getCode());
119         assertEquals("Tagihan Telah Terbayar",
120                 byrServices.prosesPembayaran
    ("332901000100100010","2013",
121                         new DateTime(2016,12,20,10,0), null).
    getMessage());
122         assertNull(byrServices.prosesPembayaran
    ("332901000100100010","2013",
123                 new DateTime(2016,12,20,10,0), null).getByrSppt
    ());
124     }
125
126     @Test
127     public void testTrxBatal() {
128         when(spDao.prosesPembayaran("332901000100100010","2013",
```

```
129                 new DateTime(2016,12,20,10,0).toDate(), null)).
    thenReturn(statusTrxBatal);
130
131         assertEquals(StatusRespond.JUMLAH_SETORAN_NIHIL,
132                 byrServices.prosesPembayaran
    ("332901000100100010","2013",
133                         new DateTime(2016,12,20,10,0), null).
    getCode());
134         assertEquals("Tagihan Telah Dibatalkan",
135                 byrServices.prosesPembayaran
    ("332901000100100010","2013",
136                         new DateTime(2016,12,20,10,0),null).
    getMessage());
137         assertNull(byrServices.prosesPembayaran
    ("332901000100100010","2013",
138                 new DateTime(2016,12,20,10,0), null).getByrSppt
    ());
139     }
140
141     @Test
142     public void testTrxError() {
143         when(spDao.prosesPembayaran("332901000100100010","2013",
144                 new DateTime(2016,12,20,10,0).toDate(), null)).
    thenReturn(statusTrxError);
145
146         assertEquals(StatusRespond.DATABASE_ERROR,
147                 byrServices.prosesPembayaran
    ("332901000100100010","2013",
148                         new DateTime(2016,12,20,10,0), null).
    getCode());
149         assertEquals("Kesalahan Server",
150                 byrServices.prosesPembayaran
    ("332901000100100010","2013",
```

```
151                                 new DateTime(2016,12,20,10,0), null).
    getMessage());
152             assertNull(byrServices.prosesPembayaran
    ("332901000100100010","2013",
153                         new DateTime(2016,12,20,10,0),null).getByrSppt()
    );
154          }
155
156 }
157
```

Seperti pada pengujian sebelumnya, pengujian kali ini pun tidak akan menggunakan data langsung dari sistem basis data, hanya dibuatkan modelnya sebagai simulasi.

Hasil dari pengujian di atas adalah seperti terlihat pada gambar 3 :



Gambar 3: Hasil *Unit Test* Untuk kelas PembayaranServicesImpl

Dari gambar di atas menunjukkan bahwa pengujian telah dilakukan dan memberikan hasil sesuai dengan yang diharapkan.

4. ReversalServicesImplTest

   Kelas `ReversalServicesImplTest` akan melakukan pengujian untuk kelas/objek `ReversalServicesImpl` yang menghubungkan antara kelas/objek `RootController` untuk fungsi reversal pembayaran. Berikut adalah kode lengkap dari kelas `ReversalServicesImplTest` :

```
1 package lab.aikibo.services;
```

```
2
3 import lab.aikibo.constant.StatusRespond;
4 import lab.aikibo.dao.StoreProceduresDaoImpl;
5 import lab.aikibo.model.ReversalPembayaran;
6 import lab.aikibo.model.StatusRev;
7 import org.junit.Before;
8 import org.junit.Test;
9 import org.junit.runner.RunWith;
10 import org.mockito.InjectMocks;
11 import org.mockito.Mock;
12 import org.springframework.test.context.junit4.SpringRunner;
13
14 import static org.junit.Assert.assertEquals;
15 import static org.junit.Assert.assertNull;
16 import static org.mockito.Mockito.when;
17
18 /**
19  * Created by tamami.
20  */
21 @RunWith(SpringRunner.class)
22 public class ReversalServicesImplTest {
23
24     @InjectMocks
25     private ReversalServicesImpl revServices = new
    ReversalServicesImpl();
26
27     @Mock
28     private StoreProceduresDaoImpl spDao;
29
30     private StatusRev statusRevBerhasil;
31     private StatusRev statusRevNihil;
32     private StatusRev statusRevGanda;
33     private StatusRev statusRevError;
```

```
34      private ReversalPembayaran revSppt;
35
36      @Before
37      public void init() {
38          revSppt = new ReversalPembayaran
    ("332901000100100010","2013","KODE_NTPD");
39          statusRevBerhasil = new StatusRev(StatusRespond.APPROVED
    , "Reversal Telah Berhasil Dilakukan", revSppt);
40          statusRevNihil = new StatusRev(StatusRespond.
    DATA_INQ_NIHIL, "Data Yang Diminta Tidak Ada",
41                  null);
42          statusRevGanda = new StatusRev(StatusRespond.
    DATABASE_ERROR, "Data Transaksi Tercatat Ganda",
43                  null);
44          statusRevError = new StatusRev(StatusRespond.
    DATABASE_ERROR, "Kesalahan Server",
45                  null);
46      }
47
48      @Test
49      public void testRevBerhasil() {
50          when(spDao.reversalPembayaran
    ("332901000100100010","2013","KODE_NTPD", null))
51                  .thenReturn(statusRevBerhasil);
52
53          assertEquals(StatusRespond.APPROVED,
54                  revServices.prosesReversal
    ("332901000100100010","2013","KODE_NTPD",null).getCode());
55          assertEquals("Reversal Telah Berhasil Dilakukan",
56                  revServices.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", null).getMessage()
    );
57          assertEquals("332901000100100010",
```

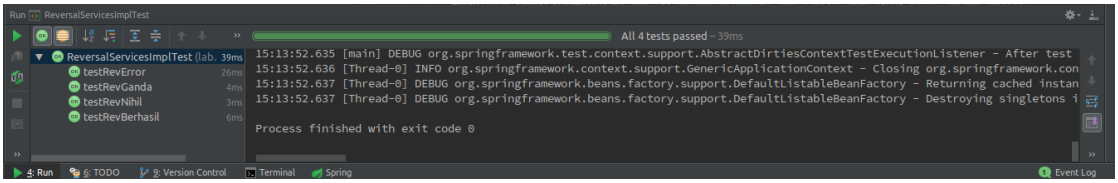
```
58                  revServices.prosesReversal
    ("332901000100100010","2013","KODE_NTPD",null)
59                          .getRevPembayaran().getNop());
60          assertEquals("2013",
61                  revServices.prosesReversal
    ("332901000100100010","2013","KODE_NTPD",null)
62                          .getRevPembayaran().getThn());
63          assertEquals("KODE_NTPD",
64                  revServices.prosesReversal
    ("332901000100100010","2013","KODE_NTPD",null)
65                          .getRevPembayaran().getNtpd());
66      }
67
68      @Test
69      public void testRevNihil() {
70          when(spDao.reversalPembayaran
    ("332901000100100010","2013","KODE_NTPD",null))
71                  .thenReturn(statusRevNihil);
72
73          assertEquals(StatusRespond.DATA_INQ_NIHIL,
74                  revServices.prosesReversal
    ("332901000100100010","2013","KODE_NTPD",null)
75                          .getCode());
76          assertEquals("Data Yang Diminta Tidak Ada",
77                  revServices.prosesReversal
    ("332901000100100010","2013","KODE_NTPD",null)
78                          .getMessage());
79          assertNull(revServices.prosesReversal
    ("332901000100100010","2013","KODE_NTPD",null)
80                  .getRevPembayaran());
81      }
82
83      @Test
```

```
84      public void testRevGanda(){
85          when(spDao.reversalPembayaran
        ("332901000100100010","2013","KODE_NTPD",null))
86                  .thenReturn(statusRevGanda);
87
88          assertEquals(StatusRespond.DATABASE_ERROR,
89                  revServices.prosesReversal
        ("332901000100100010","2013","KODE_NTPD",null)
90                          .getCode());
91          assertEquals("Data Transaksi Tercatat Ganda",
92                  revServices.prosesReversal
        ("332901000100100010","2013","KODE_NTPD",null)
93                          .getMessage());
94          assertNull(revServices.prosesReversal
        ("332901000100100010","2013","KODE_NTPD",null)
95                          .getRevPembayaran());
96      }
97
98      /**
99       * @TODO: buat unit test untuk skenario reversal gagal
        karena kesalahan DB
100      */
101     @Test
102     public void testRevError() {
103         when(spDao.reversalPembayaran
        ("332901000100100010","2013","KODE_NTPD",null))
104                 .thenReturn(statusRevError);
105
106         assertEquals(StatusRespond.DATABASE_ERROR,
107                 revServices.prosesReversal
        ("332901000100100010","2013","KODE_NTPD",null)
108                         .getCode());
109         assertEquals("Kesalahan Server",
```

```
110                     revServices.prosesReversal
    ("332901000100100010","2013","KODE_NTPD", null)
111                             .getMessage());
112         assertNull(revServices.prosesReversal
    ("332901000100100010","2013","KODE_NTPD",null)
113                             .getRevPembayaran());
114     }
115
116 }
117
```

Unit test dilakukan untuk setiap skenario yang memungkinkan terjadinya aksi, sehingga kelas ini mampu untuk melakukan tugasnya sebagai penghubung antara kelas/objek *view* dan kelas/objek *model* untuk fungsi reversal pembayaran.

Hasil dari eksekusi kode di atas adalah seperti pada gambar 4 :



Gambar 4: Hasil *Unit Test* terhadap kelas `ReversalServicesImpl`

Terlihat bahwa pengujian terhadap kelas/objek `ReversalServicesImpl` berhasil sepenuhnya.

5. SpptServicesImplTest

   Kelas/objek ini akan melakukan *unit test* pada kelas/objek *SpptServicesImpl* untuk fungsi *inquiry* data SPPT. Berikut adalah kode *unit test* untuk kelas `SpptServicesImplTest` :

```
1 package lab.aikibo.services;
```

```
2
3 import lab.aikibo.constant.StatusRespond;
4 import lab.aikibo.dao.StoreProceduresDaoImpl;
5 import lab.aikibo.model.Sppt;
6 import lab.aikibo.model.StatusInq;
7 import org.junit.Before;
8 import org.junit.Test;
9 import org.junit.runner.RunWith;
10 import org.mockito.InjectMocks;
11 import org.mockito.Mock;
12 import org.springframework.test.context.junit4.SpringRunner;
13
14 import java.math.BigInteger;
15
16 import static org.junit.Assert.assertEquals;
17 import static org.junit.Assert.assertNull;
18 import static org.mockito.Mockito.when;
19
20 /**
21  * Created by tamami.
22  */
23 @RunWith(SpringRunner.class)
24 public class SpptServicesImplTest {
25
26     @InjectMocks
27     private SpptServicesImpl spptServices = new SpptServicesImpl
      ();
28
29     @Mock
30     private StoreProceduresDaoImpl spDao;
31
32     private StatusInq statusSukses;
33     private StatusInq statusInqGagalDataTidakAda;
```

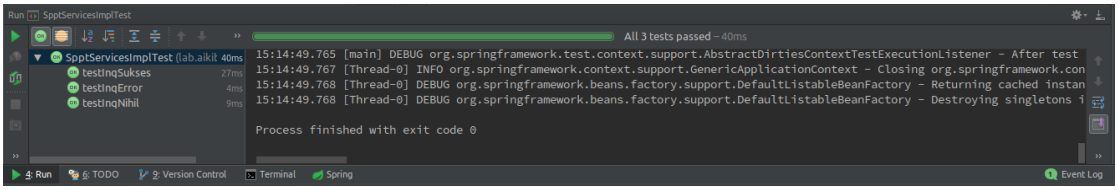
```
34      private StatusInq statusInqError;
35      private Sppt sppt;
36
37      @Before
38      public void init() {
39          sppt = new Sppt("332901000100100010", "2013", "FULAN", "
        BREBES",
40                  new BigInteger("10000"), new BigInteger("0"));
41          statusSukses = new StatusInq(1, "Data Ditemukan", sppt);
42          statusInqGagalDataTidakAda = new StatusInq(StatusRespond
        .DATA_INQ_NIHIL, "Data Tidak Ditemukan", null);
43          statusInqError = new StatusInq(StatusRespond.
        DATABASE_ERROR, "Kesalahan DB", null);
44      }
45
46      @Test
47      public void testInqSukses() {
48          when(spDao.getDataSppt("332901000100100010","2013",null)
        ).thenReturn(statusSukses);
49
50          assertEquals(StatusRespond.APPROVED,
51                  spptServices.getSpptByNopThn
        ("332901000100100010","2013",null).getCode());
52          assertEquals("Data Ditemukan",
53                  spptServices.getSpptByNopThn
        ("332901000100100010","2013",null).getMessage());
54          assertEquals("332901000100100010",
55                  spptServices.getSpptByNopThn
        ("332901000100100010","2013",null).getSppt().getNop());
56          assertEquals("2013",
57                  spptServices.getSpptByNopThn
        ("332901000100100010","2013",null).getSppt().getThn());
58          assertEquals("FULAN",
```

```
59                  spptServices.getSpptByNopThn
    ("3329010001001000 10","2013",null).getSppt().getNama());
60          assertEquals("BREBES",
61                  spptServices.getSpptByNopThn
    ("3329010001001000 10","2013",null).getSppt().getAlamatOp());
62          assertEquals(new BigInteger("10000"),
63                  spptServices.getSpptByNopThn
    ("3329010001001000 10","2013",null).getSppt().getPokok());
64          assertEquals(new BigInteger("0"),
65                  spptServices.getSpptByNopThn
    ("3329010001001000 10","2013",null).getSppt().getDenda());
66      }
67
68      /**
69       * @TODO: unit testing untuk skenario inquiry nihil
70       */
71      @Test
72      public void testInqNihil() {
73          when(spDao.getDataSppt("3329010001001000 10","2013",null)
    ).thenReturn(statusInqGagalDataTidakAda);
74
75          assertEquals(StatusRespond.DATA_INQ_NIHIL,
76                  spptServices.getSpptByNopThn
    ("3329010001001000 10","2013",null).getCode());
77          assertEquals("Data Tidak Ditemukan",
78                  spptServices.getSpptByNopThn
    ("3329010001001000 10","2013",null).getMessage());
79          assertNull(spptServices.getSpptByNopThn
    ("3329010001001000 10","2013",null).getSppt());
80      }
81
82      /**
83       * @TODO: unit testing untuk skenario inquiry error server
```

```
84        */
85      @Test
86      public void testInqError() {
87          when(spDao.getDataSppt("332901000100100010","2013",null)
    ).thenReturn(statusInqError);
88
89          assertEquals(StatusRespond.DATABASE_ERROR,
90                  spptServices.getSpptByNopThn
    ("332901000100100010","2013",null).getCode());
91          assertEquals("Kesalahan DB",
92                  spptServices.getSpptByNopThn
    ("332901000100100010","2013",null).getMessage());
93          assertNull(spptServices.getSpptByNopThn
    ("332901000100100010","2013",null).getSppt());
94      }
95
96  }
97
```

Pengujian kali ini pun menggunakan data model atau *mock* untuk memastikan bahwa proses yang terjadi menghasilkan keluaran yang diinginkan.

Hasil dari pengujian di atas adalah seperti terlihat pada gambar 5 :



Gambar 5: Hasil *Unit Test* untuk kelas `SpptServicesImpl`

Dari gambar tersebut di atas terlihat bahwa pengujian telah berhasil melakukan *unit test* pada tiap kasus.

B. *Integration Test*

Integration test akan melakukan tugasnya untuk menguji apakah koneksi dengan sistem basis data berjalan sebagaimana yang diharapkan, *integration test* yang dilakukan hanya akan dibagi menjadi 2 (dua) bagian, yaitu :

1. HibernateConfigurationIT

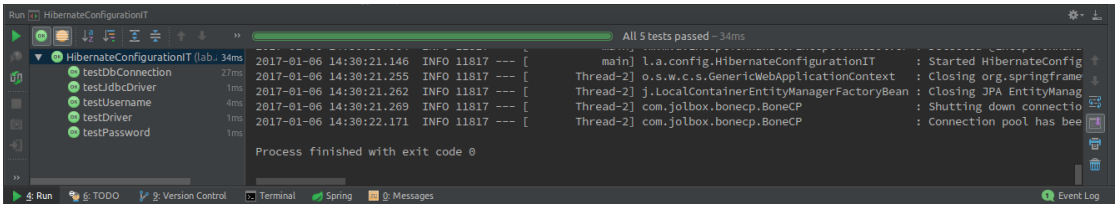
   Integration test yang dilakukan kelas ini hanya untuk memastikan parameter yang digunakan untuk melakukan koneksi ke sistem basis data sudah benar, karena parameter yang digunakan disimpan dalam bentuk *file* terpisah. Berikut adalah kode untuk melakukan integrasi terhadap *file* konfigurasi sistem basis data :

```
1  package lab.aikibo.config;
2
3  import com.jolbox.bonecp.BoneCPDataSource;
4  import lab.aikibo.App;
5  import org.junit.Test;
6  import org.junit.runner.RunWith;
7  import org.springframework.beans.factory.annotation.Autowired;
8  import org.springframework.boot.test.context.SpringBootTest;
9  import org.springframework.test.context.junit4.SpringRunner;
10
11 import static org.junit.Assert.assertEquals;
12 import static org.junit.Assert.assertNotNull;
13
14 /**
15  * Created by tamami.
16  */
17 @RunWith(SpringRunner.class)
18 @SpringBootTest(classes=App.class)
19 public class HibernateConfigurationIT {
20
```

```
21      @Autowired
22      private BoneCPDataSource boneDS;
23
24      /**
25       * check driver
26       */
27      @Test
28      public void testDriver() {
29          assertEquals("oracle.jdbc.driver.OracleDriver", boneDS.
        getDriverClass());
30      }
31
32      /**
33       * check jdbc url
34       */
35      @Test
36      public void testJdbcDriver() {
37          assertEquals("jdbc:oracle:thin:pbbdummy/pbbdummy@192
        .168.2.7:1521/sismiopbck", boneDS.getJdbcUrl());
38      }
39
40      /**
41       * check db username
42       */
43      @Test
44      public void testUsername() {
45          assertEquals("pbbdummy", boneDS.getUsername());
46      }
47
48      /**
49       * check db password
50       */
51      @Test
```

```
52      public void testPassword() {
53          assertEquals("pbbdummy", boneDS.getPassword());
54      }
55
56      /**
57       * check db connection
58       */
59      @Test
60      public void testDbConnection() {
61          assertNotNull(boneDS);
62      }
63
64  }
65
```

Hasil keluaran dari *integration test* untuk kelas `HibernateConfigurationIT` adalah seperti terlihat pada gambar 6 :



Gambar 6: Hasil *Integration Test File* Konfigurasi Sistem Basis Data

Terlihat bahwa dari 5 (lima) pengujian yang dilakukan sudah dapat memenuhi kualifikasi yang benar.

2. ServicesIT

   Kelas `ServicesIT` ini akan melakukan *integration test* terhadap kelas *services* yang hubungannya langsung melakukan aksesnya pada sistem basis data. Agar tidak mengganggu sistem yang sedang berjalan, sistem basis

data pun menggunakan model data yang sama persis sebagaimana sistem basis data yang digunakan pada fase produksi, yaitu menggunakan sistem basis data Oracle 11g.

Berikut adalah kode untuk melakukan pengujian tersebut :

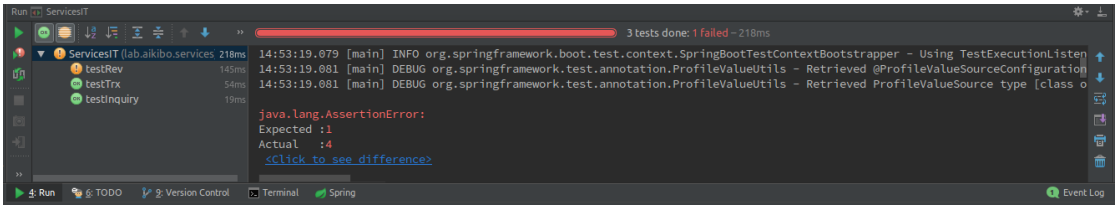
```
1  package lab.aikibo.services;
2
3  import lab.aikibo.App;
4  import lab.aikibo.constant.StatusRespond;
5  import lab.aikibo.model.StatusInq;
6  import lab.aikibo.model.StatusRev;
7  import lab.aikibo.model.StatusTrx;
8  import org.joda.time.DateTime;
9  import org.junit.Test;
10 import org.junit.runner.RunWith;
11 import org.springframework.beans.factory.annotation.Autowired;
12 import org.springframework.boot.test.context.SpringBootTest;
13 import org.springframework.test.context.junit4.SpringRunner;
14
15 import java.math.BigInteger;
16
17 import static org.junit.Assert.assertEquals;
18
19 /**
20  * Created by tamami.
21  */
22 @RunWith(SpringRunner.class)
23 @SpringBootTest(classes=App.class)
24 public class ServicesIT {
25
26     @Autowired
27     private SpptServices spptServices;
28
```

```
29      @Autowired
30      private PembayaranServices byrServices;
31
32      @Autowired
33      private ReversalServices revServices;
34
35      @Test
36      public void testInquiry() {
37          StatusInq statusInq = spptServices.getSpptByNopThn
        ("332901000100100010","2013","192.168.2.1");
38
39          assertEquals(StatusRespond.APPROVED, statusInq.getCode()
        );
40          assertEquals("Data Ditemukan", statusInq.getMessage());
41          assertEquals("332901000100100010", statusInq.getSppt().
        getNop());
42          assertEquals("2013", statusInq.getSppt().getThn());
43          assertEquals("SUKARTA", statusInq.getSppt().getNama());
44          assertEquals("GUNUNGJAYA - SALEM", statusInq.getSppt().
        getAlamatOp());
45          assertEquals(new BigInteger("35750"), statusInq.getSppt
        ().getPokok());
46          assertEquals(new BigInteger("0"), statusInq.getSppt().
        getDenda());
47      }
48
49      @Test
50      public void testTrx() {
51          StatusTrx statusTrx = byrServices.prosesPembayaran
        ("332901000100100010","2013",
52                  new DateTime(2016,12,20,10,0),null);
53
54          assertEquals(StatusRespond.APPROVED, statusTrx.getCode()
```

```
    );
55          assertEquals("Pembayaran Telah Tercatat", statusTrx.
    getMessage());
56          assertEquals("3329010001001000 10", statusTrx.getByrSppt
    ().getNop());
57          assertEquals("4.1.1.11.02", statusTrx.getByrSppt().
    getMataAnggaranPokok());
58          assertEquals(new BigInteger("35750"), statusTrx.
    getByrSppt().getPokok());
59          assertEquals("4.1.1.11.02", statusTrx.getByrSppt().
    getMataAnggaranSanksi());
60          assertEquals(new BigInteger("0"), statusTrx.getByrSppt()
    .getSanksi());
61          assertEquals("SUKARTA", statusTrx.getByrSppt().getNamaWp
    ());
62          assertEquals("GUNUNGJAYA - SALEM", statusTrx.getByrSppt
    ().getAlamatOp());
63      }
64
65      @Test
66      public void testRev() {
67          StatusRev statusRev = revServices.prosesReversal
    ("3329010001001000 10","2013",
68                  "2016AA74516SB20050812", null);
69
70          assertEquals(StatusRespond.APPROVED, statusRev.getCode()
    );
71          assertEquals("Proses Reversal Berhasil", statusRev.
    getMessage());
72          assertEquals("3329010001001000 10",statusRev.
    getRevPembayaran().getNop());
73          assertEquals("2013", statusRev.getRevPembayaran().getThn
    ());
```

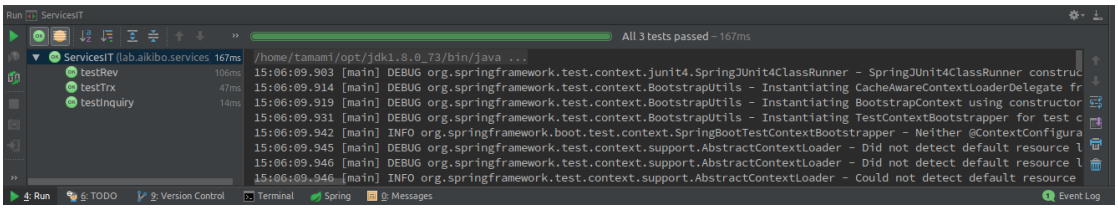
```
74          assertEquals("2016AA74516SB20050812", statusRev.
    getRevPembayaran().getNtpd());
75      }
76
77 }
78
```

Hasil dari pengujian kelas ini akan terlihat seperti pada gambar 7:



Gambar 7: Hasil *Integration Test* Dengan Sistem Basis Data

Terlihat bahwa ada kegagalan pada saat melakukan `testRev`, ini dikarenakan kode `NTPD` tidak sesuai atau tercatat pada basis data, maka diperlukan penyesuaian terhadap parameter `NTPD` yang sudah tercatat pada basis data pada pencatatan transaksi pembayaran sebelumnya, pada gambar 8 adalah hasil *integration test* ke-2 setelah dilakukan penyesuaian terhadap parameter `NTPD`.



Gambar 8: Hasil *Integration Test* Dengan Sistem Basis Data Ke-2

3 KENDALA YANG DIHADAPI

Kendala yang dihadapi pada saat melakukan *unit test* tidak ada, hanya saja pada saat melakukan *integration test*, dibutuhkan basis data model agar basis data aslinya tidak terpengaruh oleh kondisi *test* yang merubah nilai dari isi basis data.

4 KESALAHAN PROGRAM

Kesalahan program yang ditemukan selama pengujian adalah pada saat memverifikasi kode mata anggaran untuk penerimaan pajak bumi dan bangunan, keputusannya apakah penerimaan pokok pajak dipisahkan mata anggarannya dengan denda administrasi pajak daerah.

5 WAKTU PROSES UJI COBA

Untuk proses uji cobanya sendiri sangat cepat, hanya kurang dari 1 jam, karena cukup melakukan eksekusi pada seluruh unit yang ada di dalam sistem aplikasi, dan melakukan eksekusi pada beberapa unit *integration test*.