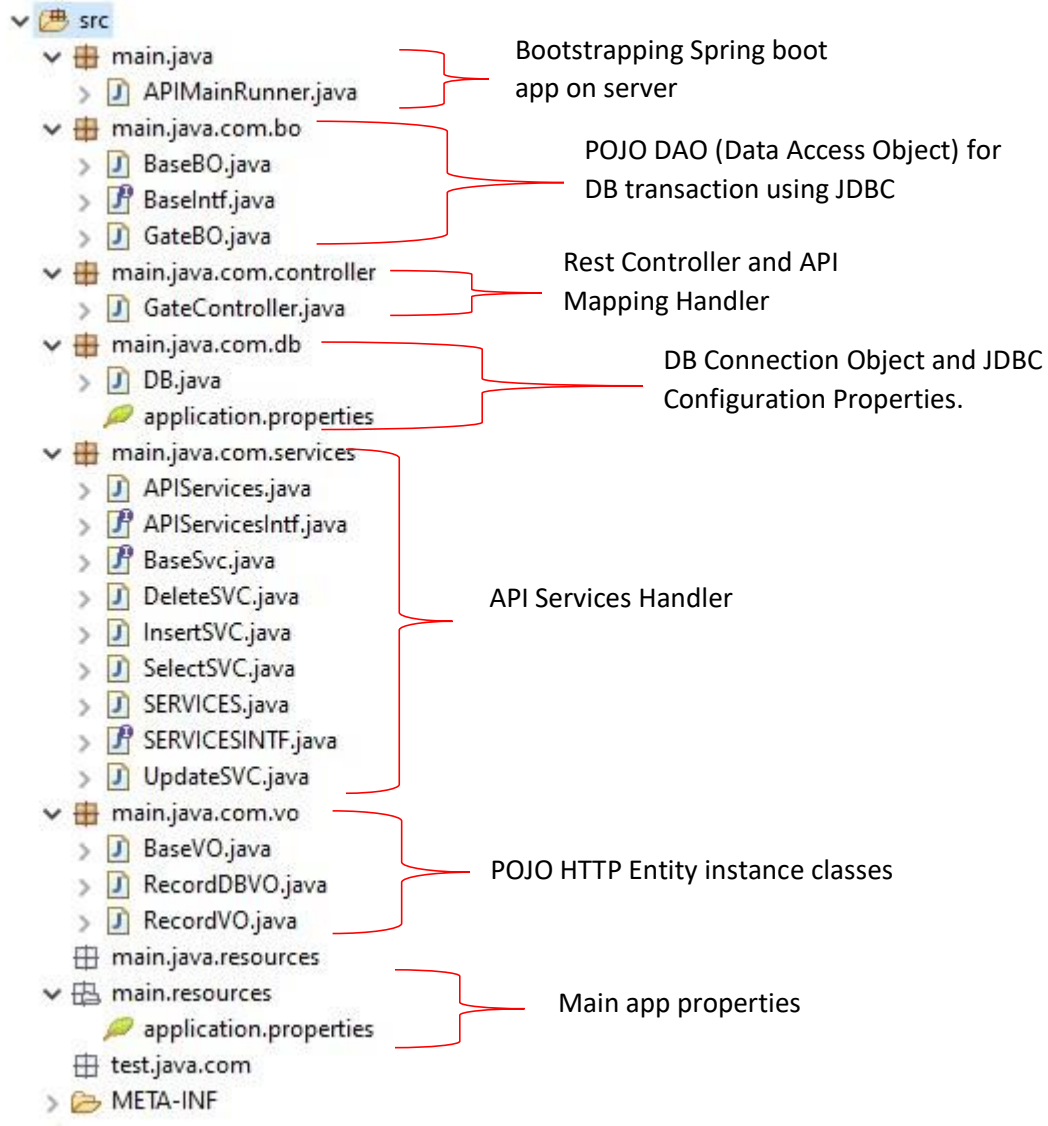
**Name: Andi Dwi Saputro**

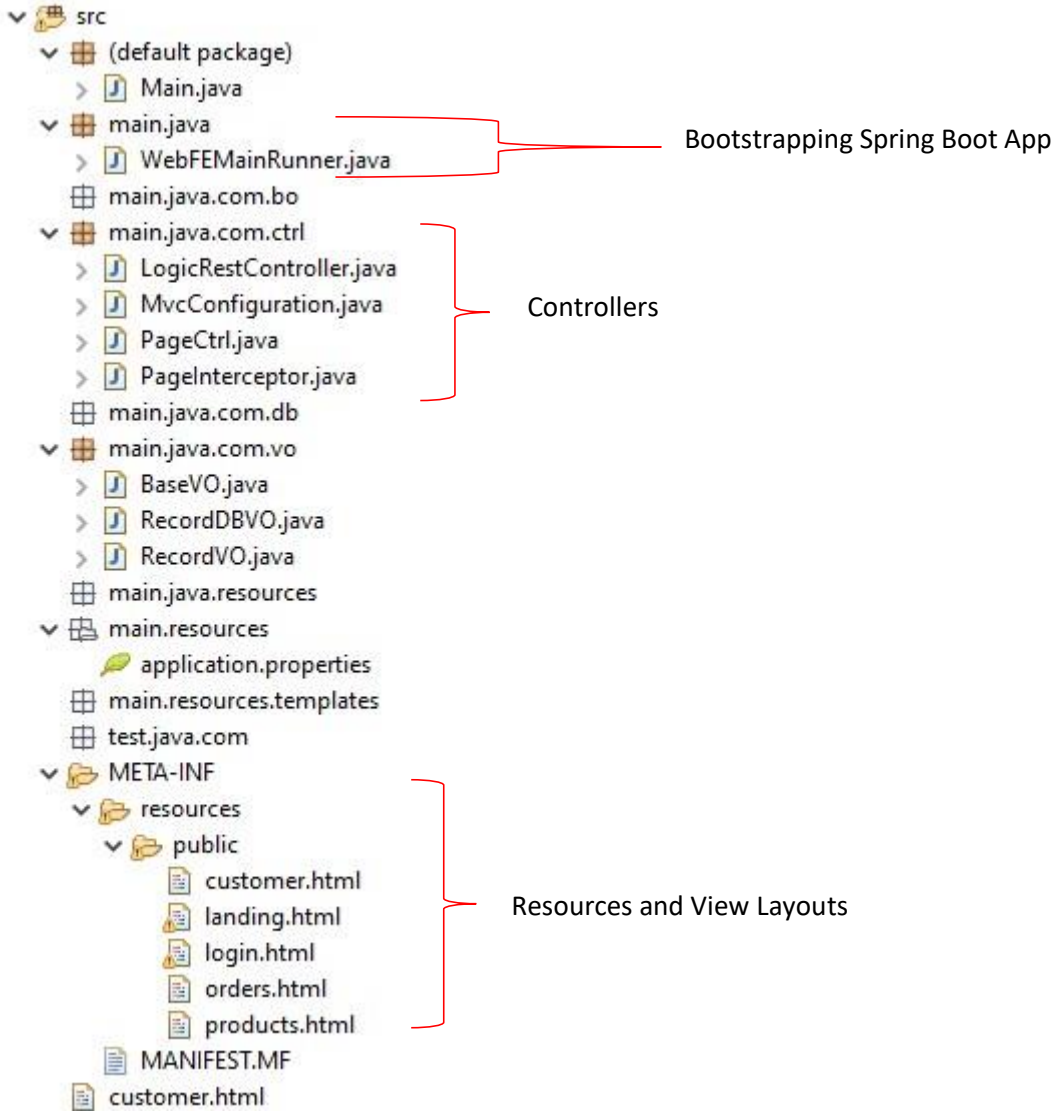
**Date: April 22nd, 2020**

---

**A. Project Structure of *quadrant_api* as a Back End/Server Side API Resources Provider Service**



**B. Project Structure of *quadrant_api_caller* as a Front End/Client Side API Resources Consumer**




1. **Initial Flow**
   a. **Initiate / Bootrapping springboot apps back end dan front end melalui file APIMainRunner.java dan WebFEMainRunner.java dengan bootstrapper annotation @SpringBootApplication**

```
7  @SpringBootApplication(exclude={DataSourceAutoConfiguration.class})
8  public class APIMainRunner {
9      public static void main(String[] args) {
10         SpringApplication.run(APIMainRunner.class, args);
11     }
```

WebFEMainRunner.java

```java
package main.java;
import org.springframework.boot.SpringApplication;

@SpringBootApplication(exclude={DataSourceAutoConfiguration.class})
public class WebFEMainRunner {
    public static void main(String[] args) {
        SpringApplication.run(WebFEMainRunner.class, args);
    }
}
```

Pada anotasi @SpringBootApplication dilakukan exclude DataSourceAutoConfiguration supaya tidak melakukan konfigurasi datasource jdbc otomatis melainkan akan dikonfigurasi manual dengan membuat object beans DB.

**b. Instansiate dan enabling Spring Boot MVC untuk front end pada file MvcConfiguration.java**

MvcConfiguration.java

```java
package main.java.com.ctrl;

import org.springframework.context.annotation.Configuration;


@Configuration
@EnableWebMvc
public class MvcConfiguration extends WebMvcConfigurerAdapter{
    @Bean
    public ViewResolver getViewResolver() {
        InternalResourceViewResolver resolver = new InternalResourceViewResolver();
        resolver.setPrefix("/public/");
        resolver.setSuffix(".html");
        return resolver;
    }

    @Bean
    PageInterceptor pageInterceptor() {
        return new PageInterceptor();
    }

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(pageInterceptor());
    }

    @Override
    public void configureDefaultServletHandling( DefaultServletHandlerConfigurer configurer) {
        configurer.enable();
    }
}
```

Code diatas merupakan custom configuration untuk spring web mvc, sejatinya pada springboot untuk mengaktifkan web mvc cukup dengan memanggil anotasi @SpringBootApplication karena anotasi tersebut sudah terdapat anotasi @EnableAutoConfiguration yang akan memanggil komponen-komponen spring-webmvc yang ada pada classpath dengan default configuration.

c. **Selesai Bootstrapping dan initiate SpringBoot Apps pada Server.**
Apabila konfigurasi selesai maka kita dapat merunning spring boot apps pada server dengn command ***mvn spring-boot:run***

2. **Flow Create (INSERT)**
   a. **Retrieve Service Command Code "SVCINS" melalui function js *_onSAVE();* dari client side (file: landing.html) ke server side setelah mengklik tombol SAVE**

**File: quadrant_api_caller/resource/public/landing.html**

```
var _onSAVE = function(){
    if(_COB!=null){
        var _cl = _COB.cols;
        var _dt = _COB.types;
        var _js = { }; var _v = []; var _r = { };
        _js["pageCD"] = _COB.pageCD; _js["svcCD"] = "SVCINS";
        for(i in _cl){
            var _val = "";
            if(_dt[i]!=='HD' && _dt[i]!=='hd') {
                _val = d3.select("#"+_cl[i]).property("value");
            }
            var _vl = _val; _v.push(_vl);
        } _js["vals"] = _v;
        _d3post._url = "/fe/ps/coded"; _d3post._js=_js; _d3post._div=""; _d3post._do(_onSAVE_do);
    }
};

var _onSAVE_do = function( _d ){
    if(_d!=null){
        alert(_d.status); _closePage("#_form");
        if(_d.status==='SUCCESS'){
            _doSELECT(_d.pageCD, "SVCSEL", "");
        }
    }
};
```

   b. **Membuat POST request dari response tombol SAVE tersebut di proses client side menggunakan RestTemplate**

**File: quadrant_api_caller/main/java/com/ctrl/LogicRestController.java**

LogicRestController.java

```
1  package main.java.com.ctrl;
2
3  import org.springframework.beans.factory.annotation.Autowired;
24
25 @RestController
26 @JsonIgnoreProperties(ignoreUnknown = true)
27 public class LogicRestController {
28     private static final String url = "http://localhost:9090/api/coded";
29     private static final String api_public_key = "quadrant_api_key";
```

Client akan mengakses resources dari end point API dari server side dengan URL ***http://localhost:9090/api/coded*** menggunakan API key: ***quadrant_api_key***

```
@RequestMapping(method = RequestMethod.POST, value = "/fe/ps/coded",
        produces = org.springframework.http.MediaType.APPLICATION_JSON_VALUE)
public @ResponseBody RecordVO callAPI(@RequestBody RecordVO vo) {
    RecordVO r = new RecordVO();
    try {
        System.out.println("pageCD = " + vo.getPageCD() + ", svcCD = " + vo.getSvcCD() );
        RestTemplate restTemplate = new RestTemplate(); vo.setApiKey( api_public_key );
        ResponseEntity<RecordVO> response = restTemplate.postForEntity(url, vo, RecordVO.class);
        r = (RecordVO) response.getBody();
    } catch (Exception e) {System.out.println("Error at LogicRestController.callAPI : " + e.getMessage() ); e.getStackTrace();}
    return r;
}
```

untuk dapat merequest akses resources tersebut client harus mengakses url ***localhost:9092/fe/ps/coded.*** Selanjutnya command service code yang sudah di retrieve dari jquery tadi **(SVCINS)** akan di kirimkan HTTP requestnya ke server side menggunakan RestTemplate dan menghasilkan result berupa response HTTP berupa JSON.

**c. Mereturn response HTTP setelah melakukan INSERT/SAVE yang diproses server side melalui API Service yang masuk ke GateController**

**File: quadrant_api/main/java/com/controller/GateController.java**

GateController.java

```
package main.java.com.controller;

import org.springframework.beans.factory.annotation.Autowired;

@RestController
@JsonIgnoreProperties(ignoreUnknown = true)
public class GateController {
   private static final String api_public_key = "quadrant_api_key";

   @Autowired
   APIServicesIntf service;

   @RequestMapping(method = RequestMethod.POST, value = "/api/coded", produces = org.springframework.http.MediaType.APPLICATION_JSON_VALUE)
   public @ResponseBody RecordVO getProduct(@RequestBody RecordVO vo) {
       if(vo!=null && vo.getApiKey() != null && vo.getApiKey().trim().equalsIgnoreCase(api_public_key)) {
           System.out.println("PAGECD = " + vo.getPageCD() + " - SVCCODE = " + vo.getSvcCD() );
           return service.runService(vo);
       }
       RecordVO r = new RecordVO(); r.setStatus("FAILED_FOR_API_KEY_WRONG");
       return r;
   }

}
```

GateController ini berfungsi sebagai mediator atau gerbang perantara setiap request HTTP yang masuk ataupun keluar ketika endpoint ***/api/coded*** diakses. GateController akan memverifikasi apakah request tersebut memiliki api key yang sama atau tidak, jika ya dan tidak kosong maka request tersebut bisa segera diproses untuk masing-masing jenis service transaction, karena tadi kita sudah meretrieve command service code **SVCINS** untuk melakukan **Insert** maka selanjutnya GateController akan mengeksekusi **InsertSVC** service yang ada pada server side ini melalui interface **APIServicesIntf**.

**File: quadrant_api/main/java/com/services/APIServicesIntf.java**

APIServicesIntf.java

```java
package main.java.com.services;

import main.java.com.vo.RecordVO;

public interface APIServicesIntf {
    public RecordVO runService(RecordVO vo);
}
```

**File: quadrant_api/main/java/com/services/InsertSVC.java**

InsertSVC.java

```java
package main.java.com.services;

import main.java.com.bo.GateBO;

public class InsertSVC extends SERVICES implements SERVICESINTF {
    private final String svcCD = "SVCINS";
    private SERVICES nextSVC = new UpdateSVC();

    @Override
    public RecordVO runService(RecordVO vo) {
        System.out.println("Before Command called: " + this.getSvcCD() + " vs " + vo.getSvcCD());
        if(vo != null && vo.getSvcCD() != null && this.getSvcCD().equalsIgnoreCase( vo.getSvcCD() ) ) {
            System.out.println("Command called: " + vo.getSvcCD());
            GateBO bo = new GateBO(); return bo.addRecord(vo);
        } else if( this.getNextSVC() != null ) {
            return this.getNextSVC().runService(vo);
        }
        return null;
    }

    @Override
    public SERVICES setNextService(SERVICES svc) {
        return this.nextSVC = svc;
    }

    public SERVICES getNextSVC() {
        return nextSVC;
    }

    public void setNextSVC(SERVICES nextSVC) {
        this.nextSVC = nextSVC;
    }

    public String getSvcCD() {
        return svcCD;
    }
}
```

Dari service Insert diatas request yang masuk akan di proses oleh Beans datasource transaction yakni **GateBO** yang akan menghandle jenis request apa yang akan di eksekusi query SQL nya. Disini SQL query yang akan di eksekusi adalah query insert dengan memanggil method ***addRecord()*** yang selanjutnya akan dieksekusi oleh **BaseBO** sebagai DAO yang akan mengeksekusi query dengan mengambil syntax query insert dari **RecordDBVO**.

**File: quadrant_api/main/com/bo/GateBO.java**

GateBO.java

```java
package main.java.com.bo;

import main.java.com.vo.RecordVO;

public class GateBO implements BaseIntf {

    @Override
    public RecordVO addRecord(RecordVO vo) {
        BaseIntf bo = new BaseBO();
        return bo.addRecord(vo);
    }

    @Override
    public RecordVO update(RecordVO vo) {
        BaseIntf bo = new BaseBO();
        return bo.update(vo);
    }

    @Override
    public RecordVO select(RecordVO vo) {
        BaseIntf bo = new BaseBO();
        return bo.select(vo);
    }

    @Override
    public RecordVO delete(RecordVO vo) {
        BaseIntf bo = new BaseBO();
        return bo.delete(vo);
    }
}
```

**File: quadrant_api/main/com/bo/BaseBO .java**

```java
    @Override
    public RecordVO addRecord(RecordVO vo) {
        try {
            RecordDBVO dbvo = getTableInfoByPageCD( vo.getPageCD() );
            if( vo != null && vo.getVals() != null && vo.getPageCD() != null && dbvo != null && dbvo.getTablename() != null) {
                DB db = new DB();

                System.out.println("add SQL: " + dbvo.getInsertSQL());
                db.openConnection();
                PreparedStatement ps = db.setSQL( dbvo.getInsertSQL() );
                String _id = db.genCid( vo.getPageCD() );
                vo.getVals().set(0, _id);
                for( int i=0; i<vo.getVals().size() - 5; i++ ) {
                    String v = vo.getVals().get(i); ps.setString((i+1), ( v != null ? v : "" ) );
                }
                int i = ps.executeUpdate();

                db.closePS(); db.closeConnection();

                fillPageAtts(vo, dbvo);

                if( i <= 0 ) { vo.setStatus(FAILED); return vo;}
            }
        }catch (Exception e) {
            System.out.println("addRecord - Exception : " + e.getMessage());
            e.getStackTrace();
            vo.setStatus(FAILED); return vo;
        }
        vo.setStatus(SUCCESS); return vo;
    }
```

**File: quadrant_api/main/com/vo/RecordDBVO.java**

```
34      public String getInsertSQL() {
35          return insertSQL;
36      }
37
38      public void setInsertSQL() {
39          if(cols!=null && cols.length > 0 && tablename!=null) {
40              this.insertSQL = "insert into " + tablename +"(";
41              String vals = "";
42              for(int i=0; i< cols.length - 5; i++ ) {
43                  String c = cols[i];
44                  this.insertSQL += c + ","; vals += "?,";
45              }
46              this.insertSQL = this.insertSQL.substring( 0, this.insertSQL.length()-1 )
47                      + ",createdby,createddate" + ") values(" + vals;
48              this.insertSQL = this.insertSQL.substring( 0, this.insertSQL.length()-1 )
49                      + insertSTMT+ ")";
50          }
51      }
```

3. **Flow Read (SELECT)**
   a. **Retrieve Service Command Code "SVCSEL" melalui fungsi js _onSELECT() dari client side ke server side setelah page load ataupun salah satu table dipilih untuk di populate datanya ke view.**

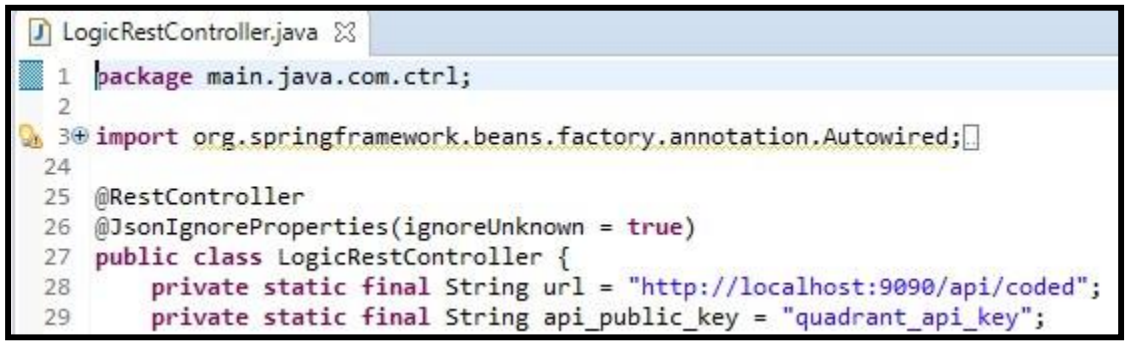
**File: quadrant_api_caller/main/resource/public/landing.html**

```
158 var _onSELECT = function(pageCD) {
159     _doSELECT(pageCD, "SVCSEL", "");
160 };
```

```
222 var  _doSELECT = function(pageCD, svcCD, _id){
223     _d3post._url = "/fe/ps/coded"; _d3post._js=_VOBJ(pageCD, svcCD, _id); _d3post._div="_vrecords"; _d3post._do(_doSELECT_do);
224 };
225
226 var _doSELECT_do = function(_d){
227     _COB = _d;
228     if(_d!=null && _d.types!=null){
229         var _colnbr = _d.types.length;
230         var _dt = _d.types;
231         var _lb = _d.labels;
232         var _lr = _d.list;
233         var _t = '<table style="width=400px;border:1px solid #000000;">';
234         _t+= '<tr><th> </th>';
235         for( i in _lb ){
236             if(_dt[i]!=='HD' && _dt[i]!=='hd') {
237                 _t+='<th style="width:120px;border:1px solid green;padding:1px;">'+ _lb[i] +'</th>';
238             }
239         }
240         _t+= '</tr>';
241         for( i in _lr ){
242             var r = _lr[i].vals;
243             _t+= '<tr>';
244             for(k in r){
245                 if(k==0){
246                     _t+='<td style="width:20px;border:1px solid green;padding:1px;"><input type="checkbox" onclick=setSELECTEDID("'+ r[k] +'") ></td>';
247                 } else
248                 if(_dt[k]!=='HD' && _dt[k]!=='hd') {
249                     _t+='<td style="width:120px;border:1px solid green;padding:1px;">'+ r[k] +'</td>';
250                 }
251             }
252             _t+='</tr>';
253         }
254         _t +='</table>';
255         d3.select("#_vrecords").html(_t);
256         d3.select("#_vlabel").html(_d.pageLabel);
257     }
258 };
```

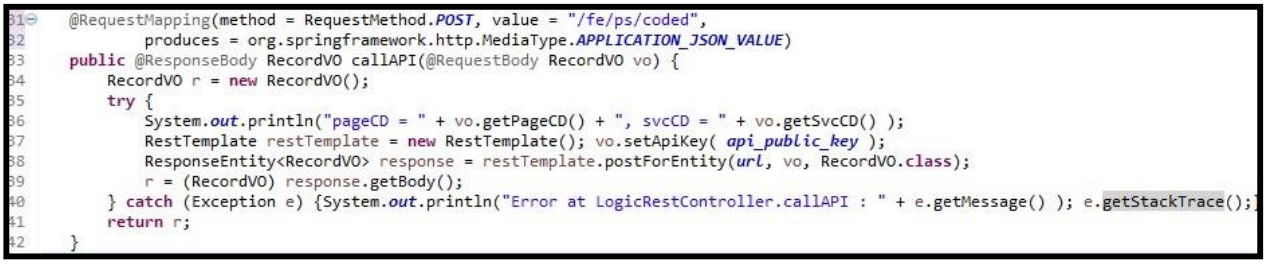
**b. Membuat POST request dari response SELECT tersebut di proses client side menggunakan RestTemplate**

**File: quadrant_api_caller/main/java/com/ctrl/LogicRestController.java**



```
LogicRestController.java
1  package main.java.com.ctrl;
2
3  import org.springframework.beans.factory.annotation.Autowired;
24
25 @RestController
26 @JsonIgnoreProperties(ignoreUnknown = true)
27 public class LogicRestController {
28     private static final String url = "http://localhost:9090/api/coded";
29     private static final String api_public_key = "quadrant_api_key";
```

Client akan mengakses resources dari end point API dari server side dengan endpoint URL ***http://localhost:9090/api/coded*** menggunakan API key: ***quadrant_api_key***



```
31  @RequestMapping(method = RequestMethod.POST, value = "/fe/ps/coded",
32          produces = org.springframework.http.MediaType.APPLICATION_JSON_VALUE)
33  public @ResponseBody RecordVO callAPI(@RequestBody RecordVO vo) {
34      RecordVO r = new RecordVO();
35      try {
36          System.out.println("pageCD = " + vo.getPageCD() + ", svcCD = " + vo.getSvcCD() );
37          RestTemplate restTemplate = new RestTemplate(); vo.setApiKey( api_public_key );
38          ResponseEntity<RecordVO> response = restTemplate.postForEntity(url, vo, RecordVO.class);
39          r = (RecordVO) response.getBody();
40      } catch (Exception e) {System.out.println("Error at LogicRestController.callAPI : " + e.getMessage() ); e.getStackTrace();}
41      return r;
42  }
```

untuk dapat merequest akses resources tersebut client harus mengakses url ***localhost:9092/fe/ps/coded.*** Selanjutnya command service code yang sudah di retrieve dari jquery tadi **(SVCSEL)** akan di kirimkan HTTP requestnya ke server side menggunakan RestTemplate dan menghasilkan result berupa response HTTP berupa JSON.

**c. Mereturn response HTTP setelah melakukan SELECT yang diproses server side melalui API Service yang masuk ke GateController**

**File: quadrant_api/main/java/com/controller/GateController.java**

```
GateController.java
1  package main.java.com.controller;
2
3  import org.springframework.beans.factory.annotation.Autowired;
25
26 @RestController
27 @JsonIgnoreProperties(ignoreUnknown = true)
28 public class GateController {
29     private static final String api_public_key = "quadrant_api_key";
30
31     @Autowired
32     APIServicesIntf service;
33
34     @RequestMapping(method = RequestMethod.POST, value = "/api/coded", produces = org.springframework.http.MediaType.APPLICATION_JSON_VALUE)
35     public @ResponseBody RecordVO getProduct(@RequestBody RecordVO vo) {
36         if(vo!=null && vo.getApiKey() != null && vo.getApiKey().trim().equalsIgnoreCase(api_public_key)) {
37             System.out.println("PAGECD = " + vo.getPageCD() + " - SVCCODE = " + vo.getSvcCD() );
38             return service.runService(vo);
39         }
40         RecordVO r = new RecordVO(); r.setStatus("FAILED_FOR_API_KEY_WRONG");
41         return r;
42     }
43
44 }
```

GateController ini berfungsi sebagai mediator atau gerbang perantara setiap request HTTP yang masuk ataupun keluar ketika endpoint ***/api/coded*** diakses. GateController akan memverifikasi apakah request tersebut memiliki api key yang sama atau tidak, jika ya dan tidak kosong maka request tersebut bisa segera diproses untuk masing-masing jenis service transaction, karena tadi kita sudah meretrieve command service code **SVCSEL** untuk melakukan **SELECT** maka selanjutnya GateController akan mengeksekusi **SelectSVC** service yang ada pada server side ini melalui interface **APIServicesIntf**.

**File: quadrant_api/main/java/com/services/APIServicesIntf.java**

APIServicesIntf.java

```
1  package main.java.com.services;
2
3  import main.java.com.vo.RecordVO;
4
5  public interface APIServicesIntf {
6      public RecordVO runService(RecordVO vo);
7  }
```

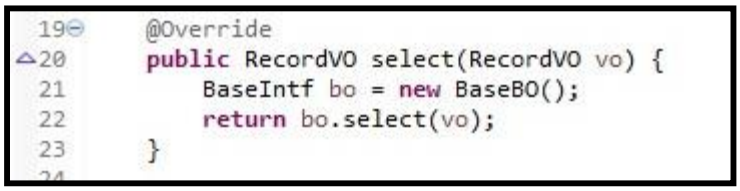
**File: quadrant_api/main/java/com/services/SelectSVC.java**

SelectSVC.java

```
1  package main.java.com.services;
2
3  import main.java.com.bo.GateBO;
5
6  public class SelectSVC extends SERVICES implements SERVICESINTF {
7      private final String svcCD = "SVCSEL";
8      private SERVICES nextSVC = new DeleteSVC();
9
10     @Override
11     public RecordVO runService(RecordVO vo) {
12         System.out.println("Before Command called: " + this.getSvcCD() + " vs " + vo.getSvcCD());
13         if(vo != null && vo.getSvcCD() != null && this.getSvcCD().equalsIgnoreCase( vo.getSvcCD() ) ) {
14             System.out.println("Command called: " + vo.getSvcCD());
15             GateBO bo = new GateBO(); return bo.select(vo);
16         } else if( this.getNextSVC() != null ) {
17             return this.getNextSVC().runService(vo);
18         }
19         return null;
20     }
21
22     @Override
23     public SERVICES setNextService(SERVICES svc) {
24         return this.nextSVC = svc;
25     }
26
27     public SERVICES getNextSVC() {
28         return nextSVC;
29     }
30
31     public void setNextSVC(SERVICES nextSVC) {
32         this.nextSVC = nextSVC;
33     }
34
35     public String getSvcCD() {
36         return svcCD;
37     }
38 }
```

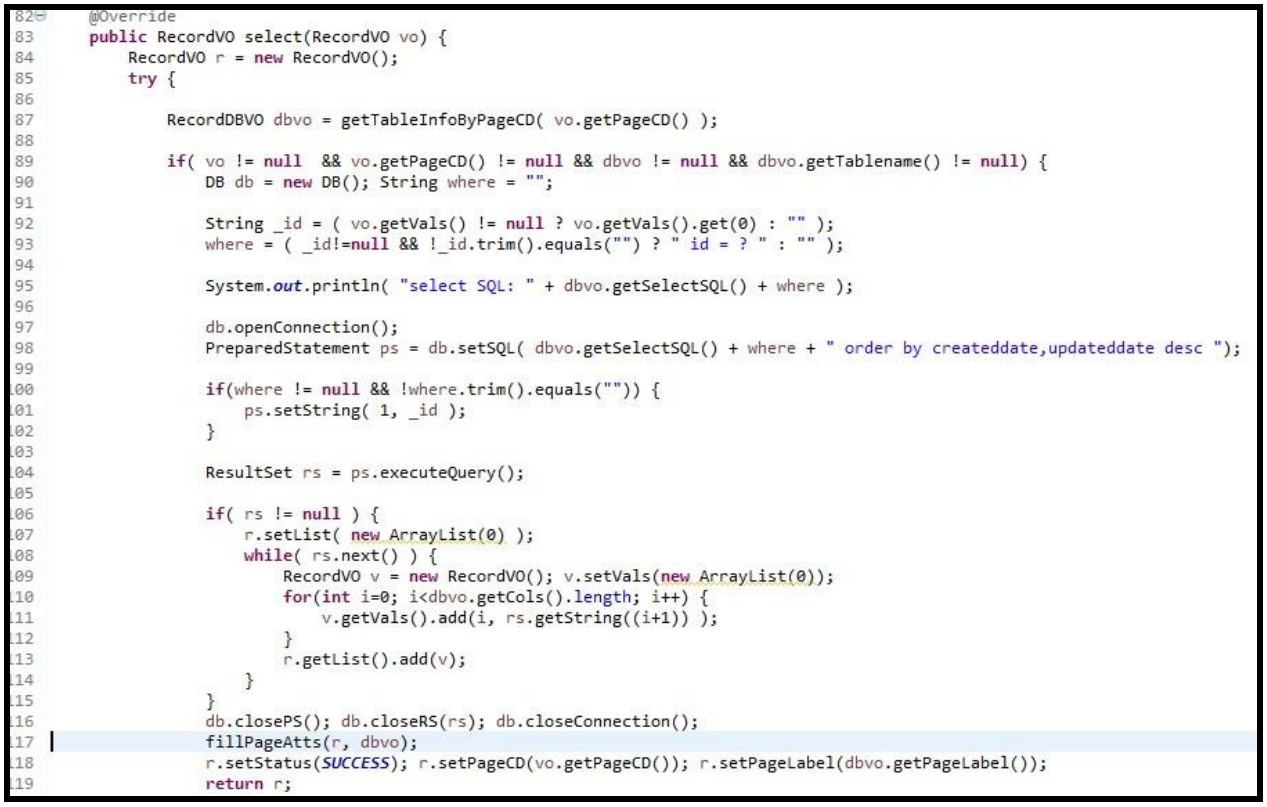
Dari service Select diatas request yang masuk akan di proses oleh Beans datasource transaction yakni **GateBO** yang akan menghandle jenis request apa yang akan di eksekusi query SQL nya. Disini SQL query yang akan di eksekusi adalah query select dengan memanggil method ***select()*** yang selanjutnya akan dieksekusi oleh **BaseBO** sebagai DAO yang akan mengeksekusi query dengan mengambil syntax query select dari **RecordDBVO**.

**File: quadrant_api/main/com/bo/GateBO.java**



```
19      @Override
20      public RecordVO select(RecordVO vo) {
21          BaseIntf bo = new BaseBO();
22          return bo.select(vo);
23      }
```

**File: quadrant_api/main/com/bo/BaseBO .java**



```
82      @Override
83      public RecordVO select(RecordVO vo) {
84          RecordVO r = new RecordVO();
85          try {
86
87              RecordDBVO dbvo = getTableInfoByPageCD( vo.getPageCD() );
88
89              if( vo != null  && vo.getPageCD() != null && dbvo != null && dbvo.getTablename() != null) {
90                  DB db = new DB(); String where = "";
91
92                  String _id = ( vo.getVals() != null ? vo.getVals().get(0) : "" );
93                  where = ( _id!=null && !_id.trim().equals("") ? " id = ? " : "" );
94
95                  System.out.println( "select SQL: " + dbvo.getSelectSQL() + where );
96
97                  db.openConnection();
98                  PreparedStatement ps = db.setSQL( dbvo.getSelectSQL() + where + " order by createddate,updateddate desc ");
99
100                 if(where != null && !where.trim().equals("")) {
101                     ps.setString( 1, _id );
102                 }
103
104                 ResultSet rs = ps.executeQuery();
105
106                 if( rs != null ) {
107                     r.setList( new ArrayList(0) );
108                     while( rs.next() ) {
109                         RecordVO v = new RecordVO(); v.setVals(new ArrayList(0));
110                         for(int i=0; i<dbvo.getCols().length; i++) {
111                             v.getVals().add(i, rs.getString((i+1)) );
112                         }
113                         r.getList().add(v);
114                     }
115                 }
116                 db.closePS(); db.closeRS(rs); db.closeConnection();
117                 fillPageAtts(r, dbvo);
118                 r.setStatus(SUCCESS); r.setPageCD(vo.getPageCD()); r.setPageLabel(dbvo.getPageLabel());
119                 return r;
```

**File: quadrant_api/main/com/vo/RecordDBVO.java**

```
53    public String getSelectSQL() {
54        return selectSQL;
55    }
56    public void setSelectSQL() {
57        if(cols!=null && cols.length > 0 && tablename!=null) {
58            this.selectSQL = "select ";
59            for(String c: cols) {
60                this.selectSQL += c + ",";
61            }
62            this.selectSQL = this.selectSQL.substring( 0, this.selectSQL.length()-1 ) + " from " + tablename;
63        }
64    }
```

4. **Flow Update (UPDATE)**
   a. **Retrieve Service Command Code "SVCUPD" melalui fungsi js _onEDIT() dari client side ke server side setelah klik tombol EDIT.**

**File: quadrant_api_caller/main/resources/public/landing.html**

```
162 var _onEDIT = function(pageCD) {
163     if( __selectedId != null && __selectedId !== '' ){
164         var _cl = _COB.cols;
165         var _js = { }; var _v = []; var _r = { };
166         _js["pageCD"] = _COB.pageCD; _js["svcCD"] = "SVCUPD";
167         for(i in _cl){
168             var _val = ''; _v.push(_val); /*d3.select("#"+_cl[i]).property("value");*/
169         } _v[0] = __selectedId; _js["vals"] = _v;
170         alert(JSON.stringify(_js));
171         _d3post._url = "/fe/ps/coded"; _d3post._js=_js; _d3post._div=""; _d3post._do(_onEDIT_do);
172     }
173 };
174
175 var _onEDIT_do = function(_d){
176     if(_d!=null && _d.list!=null && _d.list.length >= 1){
177         var _ar = _d.list[0];
178         alert( JSON.stringify(_ar) );
179     } else {
180         alert("Record not found.");
181     }
182 };
```

   b. **Membuat POST request dari response UPDATE tersebut di proses client side menggunakan RestTemplate**

**File: quadrant_api_caller/main/java/com/ctrl/LogicRestController.java**

LogicRestController.java

```
1  package main.java.com.ctrl;
2
3  import org.springframework.beans.factory.annotation.Autowired;
24
25 @RestController
26 @JsonIgnoreProperties(ignoreUnknown = true)
27 public class LogicRestController {
28     private static final String url = "http://localhost:9090/api/coded";
29     private static final String api_public_key = "quadrant_api_key";
```

Client akan mengakses resources dari end point API dari server side dengan endpoint URL ***http://localhost:9090/api/coded*** menggunakan API key: ***quadrant_api_key***

```
@RequestMapping(method = RequestMethod.POST, value = "/fe/ps/coded",
        produces = org.springframework.http.MediaType.APPLICATION_JSON_VALUE)
public @ResponseBody RecordVO callAPI(@RequestBody RecordVO vo) {
    RecordVO r = new RecordVO();
    try {
        System.out.println("pageCD = " + vo.getPageCD() + ", svcCD = " + vo.getSvcCD() );
        RestTemplate restTemplate = new RestTemplate(); vo.setApiKey( api_public_key );
        ResponseEntity<RecordVO> response = restTemplate.postForEntity(url, vo, RecordVO.class);
        r = (RecordVO) response.getBody();
    } catch (Exception e) {System.out.println("Error at LogicRestController.callAPI : " + e.getMessage() ); e.getStackTrace();}
    return r;
}
```

untuk dapat merequest akses resources tersebut client harus mengakses url ***localhost:9092/fe/ps/coded.*** Selanjutnya command service code yang sudah di retrieve dari jquery tadi **(SVCUPD)** akan di kirimkan HTTP requestnya ke server side menggunakan RestTemplate dan menghasilkan result berupa response HTTP berupa JSON.

**c. Mereturn response HTTP setelah melakukan UPDATE yang diproses server side melalui API Service yang masuk ke GateController**

**File: quadrant_api/main/java/com/controller/GateController.java**

GateController.java

```
package main.java.com.controller;

import org.springframework.beans.factory.annotation.Autowired;

@RestController
@JsonIgnoreProperties(ignoreUnknown = true)
public class GateController {
    private static final String api_public_key = "quadrant_api_key";

    @Autowired
    APIServicesIntf service;

    @RequestMapping(method = RequestMethod.POST, value = "/api/coded", produces = org.springframework.http.MediaType.APPLICATION_JSON_VALUE)
    public @ResponseBody RecordVO getProduct(@RequestBody RecordVO vo) {
        if(vo!=null && vo.getApiKey() != null && vo.getApiKey().trim().equalsIgnoreCase(api_public_key)) {
            System.out.println("PAGECD = " + vo.getPageCD() + " - SVCCODE = " + vo.getSvcCD() );
            return service.runService(vo);
        }
        RecordVO r = new RecordVO(); r.setStatus("FAILED_FOR_API_KEY_WRONG");
        return r;
    }

}
```

GateController ini berfungsi sebagai mediator atau gerbang perantara setiap request HTTP yang masuk ataupun keluar ketika endpoint ***/api/coded*** diakses. GateController akan memverifikasi apakah request tersebut memiliki api key yang sama atau tidak, jika ya dan tidak kosong maka request tersebut bisa segera diproses untuk masing-masing jenis service transaction, karena tadi kita sudah meretrieve command service code **SVCUPD** untuk melakukan **UPDATE** maka selanjutnya GateController akan mengeksekusi **UpdateSVC** service yang ada pada server side ini melalui interface **APIServicesIntf**.

**File: quadrant_api/main/java/com/services/APIServicesIntf.java**

APIServicesIntf.java

```java
package main.java.com.services;

import main.java.com.vo.RecordVO;

public interface APIServicesIntf {
    public RecordVO runService(RecordVO vo);
}
```

**File: quadrant_api/main/java/com/services/UpdateSVC.java**

UpdateSVC.java

```java
package main.java.com.services;

import main.java.com.bo.GateBO;

public class UpdateSVC extends SERVICES implements SERVICESINTF {
    private final String svcCD = "SVCUPD";
    private SERVICES nextSVC = new SelectSVC();

    @Override
    public RecordVO runService(RecordVO vo) {
        System.out.println("Before Command called: " + this.getSvcCD() + " vs " + vo.getSvcCD());
        if(vo != null && vo.getSvcCD() != null && this.getSvcCD().equalsIgnoreCase( vo.getSvcCD() ) ) {
            System.out.println("Command called: " + vo.getSvcCD());
            GateBO bo = new GateBO(); return bo.update(vo);
        } else if( this.getNextSVC() != null ) {
            return this.getNextSVC().runService(vo);
        }
        return null;
    }

    @Override
    public SERVICES setNextService(SERVICES svc) {
        return this.nextSVC = svc;
    }

    public SERVICES getNextSVC() {
        return nextSVC;
    }

    public void setNextSVC(SERVICES nextSVC) {
        this.nextSVC = nextSVC;
    }

    public String getSvcCD() {
        return svcCD;
    }

}
```

Dari service Update diatas request yang masuk akan di proses oleh Beans datasource transaction yakni **GateBO** yang akan menghandle jenis request apa yang akan di eksekusi query SQL nya. Disini SQL query yang akan di eksekusi adalah query update dengan memanggil method ***update()*** yang selanjutnya akan dieksekusi oleh **BaseBO** sebagai DAO yang akan mengeksekusi query dengan mengambil syntax query update dari **RecordDBVO**.

**File: quadrant_api/main/com/bo/GateBO.java**

```
13      @Override
14      public RecordVO update(RecordVO vo) {
15          BaseIntf bo = new BaseBO();
16          return bo.update(vo);
17      }
18
```

**File: quadrant_api/main/com/bo/BaseBO .java**

```
138     @Override
139     public RecordVO update(RecordVO vo) {
140         try {
141             RecordDBVO dbvo = getTableInfoByPageCD( vo.getPageCD() );
142             if( vo != null && vo.getVals() != null && vo.getPageCD() != null && dbvo != null && dbvo.getTablename() != null) {
143                 DB db = new DB();
144
145                 System.out.println("update SQL: " + dbvo.getUpdateSQL());
146
147                 db.openConnection();
148                 PreparedStatement ps = db.setSQL( dbvo.getUpdateSQL() );
149
150                 String _id = vo.getVals().get(0); int j=0;
151
152                 for( int i=1; i < vo.getVals().size() - 5 ; i++ ) {
153                     String v = vo.getVals().get(i); ps.setString((i+1), ( v != null ? v : "" ) ); j=(i+1);
154                 }
155
156                 ps.setString(++j, _id);
157
158                 int i = ps.executeUpdate();
159
160                 db.closePS(); db.closeConnection();
161                 fillPageAtts(vo, dbvo);
162                 if( i <= 0 ) { vo.setStatus(FAILED); return vo; }
163
164             }
165         }catch (Exception e) {
166             System.out.println("update - Exception : "+ e.getMessage());
167             e.getStackTrace();
168             vo.setStatus(FAILED); return vo;
169         }
170         vo.setStatus(SUCCESS); vo.setVals(null);
171         return vo;
172     }
```

**File: quadrant_api/main/com/vo/RecordDBVO.java**

```
19      public String getUpdateSQL() {
20          return updateSQL;
21      }
22      public void setUpdateSQL() {
23          if(cols!=null && cols.length > 0 && tablename!=null) {
24              this.updateSQL = "update " + tablename +" set "; String vals = "";
25              for(int i=0; i< cols.length - 5; i++) {
26                  String c = cols[i];
27                  if( i == 0 ) continue;
28                  this.updateSQL += c + "=?,";
29              }
30              this.updateSQL = this.updateSQL.substring( 0, this.updateSQL.length()-1 ) + updateSTMT + " where id=? ";
31          }
32      }
```

5. **Flow Delete (DELETE)**
   a. **Retrieve Service Command Code "SVCDEL" melalui fungsi js _onDELETE() dari client side ke server side setelah klik tombol DELETE.**

      **File: quadrant_api_caller/main/resources/landing.html**

```
184 var _onDELETE = function(pageCD) {
185     if ( __selectedId != null && __selectedId !== '' ){
186         var _cl = _COB.cols;
187         var _js = { }; var _v = []; var _r = { };
188         _js["pageCD"] = _COB.pageCD; _js["svcCD"] = "SVCDEL";
189         for (i in _cl){
190             var _val = ''; _v.push(_val)
191         } _v[0] = __selectedId; _js["vals"] = _v;
192         alert(JSON.stringify(_js));
193         _d3post._url = "/fe/ps/coded"; _d3post._js=_js; _d3post._div=""; _d3post._do(_onDELETE_do);
194     }
195 };
196
197 var _onDELETE_do = function(_d){
198     if (_d!=null && _d.list != null && _d.list.length >= 1){
199         var _ar = _d.list[0];
200         alert( JSON.stringify(_ar) )
201     } else {
202         alert("Record not found.");
203     }
204 };
```

      **** fungsi onDelete pada file originallnya belum ada codenya saya coba tambahkan sendiri dengan mengikuti dari fungsi onEDIT karena sama-sama menselect Id.***

   b. **Membuat POST request dari response DELETE tersebut di proses client side menggunakan RestTemplate**

      **File: quadrant_api_caller/main/java/com/ctrl/LogicRestController.java**

```
LogicRestController.java
 1 package main.java.com.ctrl;
 2
 3 import org.springframework.beans.factory.annotation.Autowired;
24
25 @RestController
26 @JsonIgnoreProperties(ignoreUnknown = true)
27 public class LogicRestController {
28     private static final String url = "http://localhost:9090/api/coded";
29     private static final String api_public_key = "quadrant_api_key";
```

      Client akan mengakses resources dari end point API dari server side dengan endpoint URL ***http://localhost:9090/api/coded*** menggunakan API key: ***quadrant_api_key***

```
@RequestMapping(method = RequestMethod.POST, value = "/fe/ps/coded",
        produces = org.springframework.http.MediaType.APPLICATION_JSON_VALUE)
public @ResponseBody RecordVO callAPI(@RequestBody RecordVO vo) {
    RecordVO r = new RecordVO();
    try {
        System.out.println("pageCD = " + vo.getPageCD() + ", svcCD = " + vo.getSvcCD() );
        RestTemplate restTemplate = new RestTemplate(); vo.setApiKey( api_public_key );
        ResponseEntity<RecordVO> response = restTemplate.postForEntity(url, vo, RecordVO.class);
        r = (RecordVO) response.getBody();
    } catch (Exception e) {System.out.println("Error at LogicRestController.callAPI : " + e.getMessage() ); e.getStackTrace();}
    return r;
}
```

untuk dapat merequest akses resources tersebut client harus mengakses url ***localhost:9092/fe/ps/coded.*** Selanjutnya command service code yang sudah di retrieve dari jquery tadi **(SVCDEL)** akan di kirimkan HTTP requestnya ke server side menggunakan RestTemplate dan menghasilkan result berupa response HTTP berupa JSON.

**c. Mereturn response HTTP setelah melakukan DELETE yang diproses server side melalui API Service yang masuk ke GateController**

**File: quadrant_api/main/java/com/controller/GateController.java**

GateController.java

```
package main.java.com.controller;

import org.springframework.beans.factory.annotation.Autowired;

@RestController
@JsonIgnoreProperties(ignoreUnknown = true)
public class GateController {
    private static final String api_public_key = "quadrant_api_key";

    @Autowired
    APIServicesIntf service;

    @RequestMapping(method = RequestMethod.POST, value = "/api/coded", produces = org.springframework.http.MediaType.APPLICATION_JSON_VALUE)
    public @ResponseBody RecordVO getProduct(@RequestBody RecordVO vo) {
        if(vo!=null && vo.getApiKey() != null && vo.getApiKey().trim().equalsIgnoreCase(api_public_key)) {
            System.out.println("PAGECD = " + vo.getPageCD() + " - SVCCODE = " + vo.getSvcCD() );
            return service.runService(vo);
        }
        RecordVO r = new RecordVO(); r.setStatus("FAILED_FOR_API_KEY_WRONG");
        return r;
    }

}
```

GateController ini berfungsi sebagai mediator atau gerbang perantara setiap request HTTP yang masuk ataupun keluar ketika endpoint ***/api/coded*** diakses. GateController akan memverifikasi apakah request tersebut memiliki api key yang sama atau tidak, jika ya dan tidak kosong maka request tersebut bisa segera diproses untuk masing-masing jenis service transaction, karena tadi kita sudah meretrieve command service code **SVCDEL** untuk melakukan **DELETE** maka selanjutnya GateController akan mengeksekusi **DeleteSVC** service yang ada pada server side ini melalui interface **APIServicesIntf**.

**File: quadrant_api/main/java/com/services/APIServicesIntf.java**

APIServicesIntf.java

```java
package main.java.com.services;

import main.java.com.vo.RecordVO;

public interface APIServicesIntf {
    public RecordVO runService(RecordVO vo);
}
```

**File: quadrant_api/main/java/com/services/DeleteSVC.java**

DeleteSVC.java

```java
package main.java.com.services;

import main.java.com.bo.GateBO;
import main.java.com.vo.RecordVO;

public class DeleteSVC extends SERVICES implements SERVICESINTF {
    private final String svcCD = "SVCDEL";
    private SERVICES nextSVC = null;

    @Override
    public RecordVO runService(RecordVO vo) {
        System.out.println("Before Command called: " + this.getSvcCD() + " vs " + vo.getSvcCD());
        if(vo != null && vo.getSvcCD() != null && this.getSvcCD().equalsIgnoreCase( vo.getSvcCD() ) ) {
            System.out.println("Command called: " + vo.getSvcCD());
            GateBO bo = new GateBO(); return bo.delete(vo);
        } else if( this.getNextSVC() != null ) {
            return this.getNextSVC().runService(vo);
        }
        return null;
    }

    @Override
    public SERVICES setNextService(SERVICES svc) {
        return this.nextSVC = svc;
    }

    public SERVICES getNextSVC() {
        return nextSVC;
    }

    public void setNextSVC(SERVICES nextSVC) {
        this.nextSVC = nextSVC;
    }

    public String getSvcCD() {
        return svcCD;
    }
}
```

Dari service Delete diatas request yang masuk akan di proses oleh Beans datasource transaction yakni **GateBO** yang akan menghandle jenis request apa yang akan di eksekusi query SQL nya. Disini SQL query yang akan di eksekusi adalah query update dengan memanggil method ***delete()*** yang selanjutnya akan dieksekusi oleh **BaseBO** sebagai DAO yang akan mengeksekusi query dengan mengambil syntax query delete dari **RecordDBVO**.

**File: quadrant_api/main/com/bo/GateBO.java**

```
25      @Override
26      public RecordVO delete(RecordVO vo) {
27          BaseIntf bo = new BaseBO();
28          return bo.delete(vo);
29      }
30  }
```

**File: quadrant_api/main/com/bo/BaseBO .java**

```
174     @Override
175     public RecordVO delete(RecordVO vo) {
176         try {
177             RecordDBVO dbvo = getTableInfoByPageCD( vo.getPageCD() );
178             if( vo != null && vo.getVals() != null && vo.getPageCD() != null && dbvo != null && dbvo.getTablename() != null) {
179                 DB db = new DB();
180                 System.out.println("delete SQL: " + dbvo.getDeleteSQL() );
181                 db.openConnection();
182                 PreparedStatement ps = db.setSQL( dbvo.getDeleteSQL() );
183                 String _id = vo.getVals().get(0);
184                 ps.setString(1, _id);
185                 int i = ps.executeUpdate();
186                 db.closePS(); db.closeConnection();
187                 fillPageAtts(vo, dbvo);
188                 if( i <= 0 ) { vo.setStatus(FAILED); return vo;}
189             }
190         }catch (Exception e) {
191             System.out.println("delete - Exception : "+ e.getMessage());
192             e.getStackTrace();
193             vo.setStatus(FAILED); return vo;
194         }
195         vo.setStatus(SUCCESS); vo.setVals(null);
196         return vo;
197     }
198 }
```

**File: quadrant_api/main/com/vo/RecordDBVO.java**

```
66      public String getDeleteSQL() {
67          return deleteSQL;
68      }
69
70      public void setDeleteSQL() {
71          if(cols!=null && cols.length > 0 && tablename!=null) {
72              this.deleteSQL = "update " + tablename + " set delflag = 'Y' where id = ?";
73          }
74      }
```