**Name: Andi Dwi Saputro**

**Date: April 22nd, 2020**

1. **Project Structure of *quadrant\_api* as a Back End/Server Side API Resources Provider Service**

POJO DAO (Data Access Object) for

DB transaction using JDBC

Main app properties

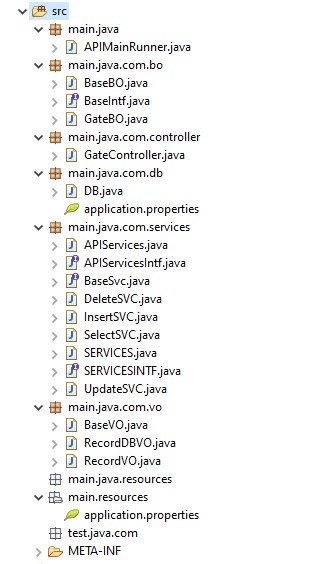
POJO HTTP Entity instance classes

API Services Handler

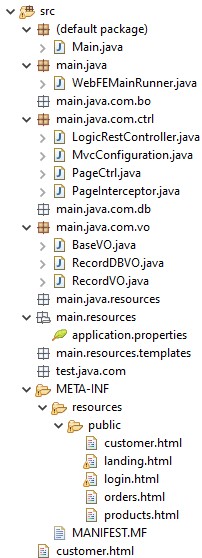
DB Connection Object and JDBC Configuration Properties.

Rest Controller and API Mapping Handler

Bootstrapping Spring boot app on server



1. **Project Structure of *quadrant\_api\_caller* as a Front End/Client Side API Resources Consumer**

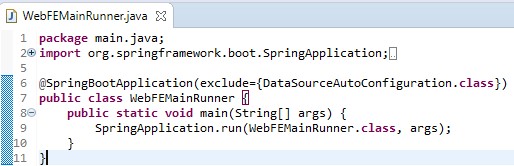
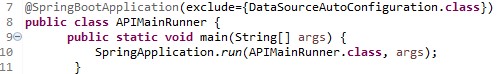
****

Resources and View Layouts

Controllers

Bootstrapping Spring Boot App

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



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

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



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.

1. **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***

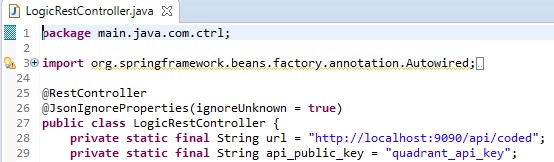
1. **Flow Create (INSERT)**
2. **Retrieve Service Command Code “SVCINS” melalui function js *\_onSAVE();* dari client side (file: landing.html) ke server side setelah mengklik tombol SAVE**

**File: quadrant\_api\_caller/resource/public/landing.html**

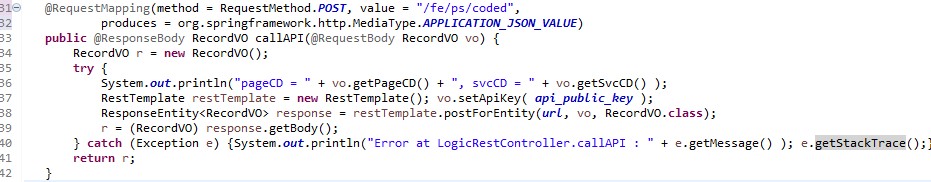


1. **Membuat POST request dari response tombol SAVE tersebut di proses client side menggunakan RestTemplate**

**File: quadrant\_api\_caller/main/java/com/ctrl/LogicRestController.java**



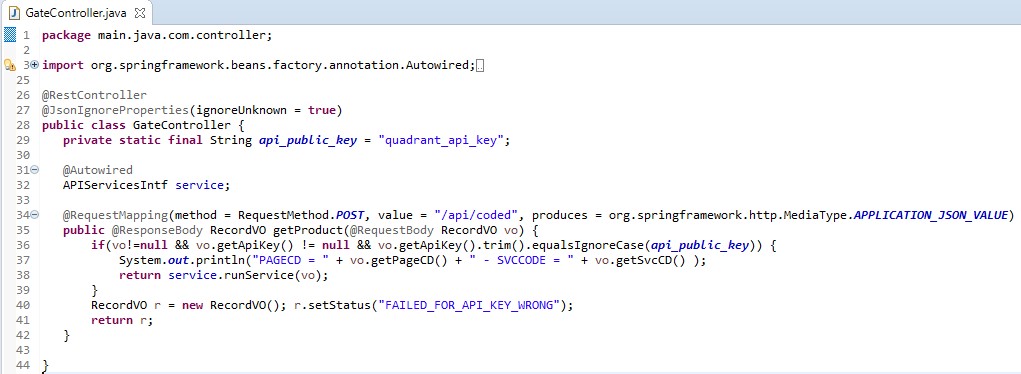
Client akan mengakses resources dari end point API dari server side dengan URL [***http://localhost:9090/api/coded***](http://localhost:9090/api/coded) menggunakan API key: ***quadrant\_api\_key***



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.

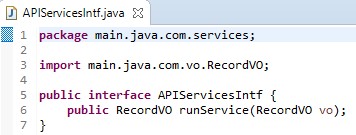
1. **Mereturn response HTTP setelah melakukan INSERT/SAVE yang diproses server side melalui API Service yang masuk ke GateController**

**File: quadrant\_api/main/java/com/controller/GateController.java**

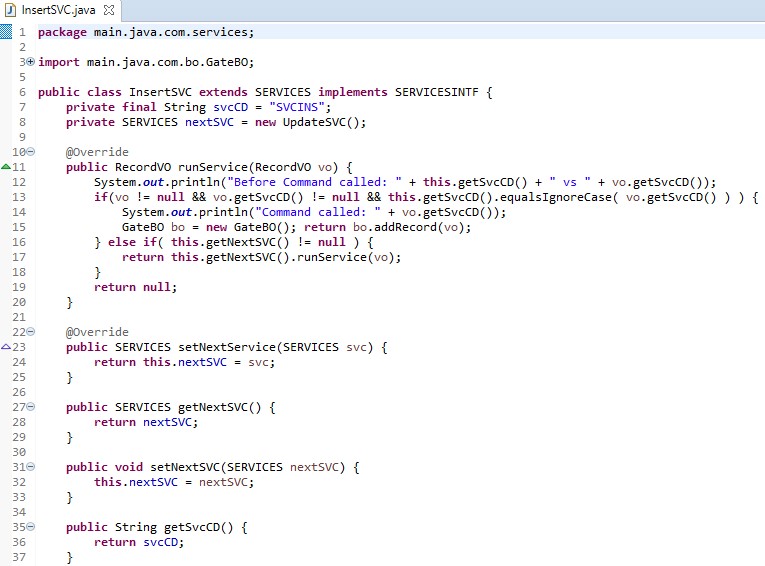


GateController ini berfungsi sebagai mediator atau gerbang perantara setiap request HTTP yang masuk ataupun keluar ketika endpoint ***/api/coded*** diakses. GateController akan memverifikasi apakah request tersebut memiliki api key yang sama atau tidak, jika ya dan tidak kosong maka request tersebut 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**

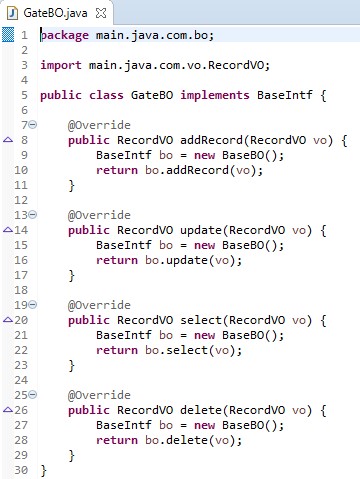


**File: quadrant\_api/main/java/com/services/InsertSVC.java**



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**.

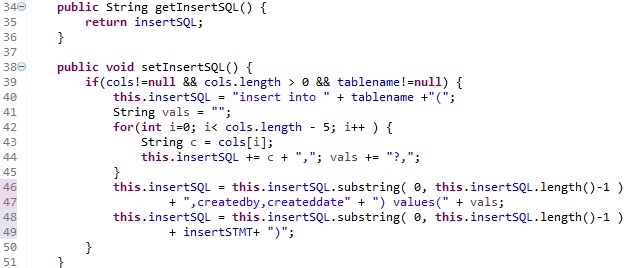
**File: quadrant\_api/main/com/bo/GateBO.java**



**File: quadrant\_api/main/com/bo/BaseBO .java**

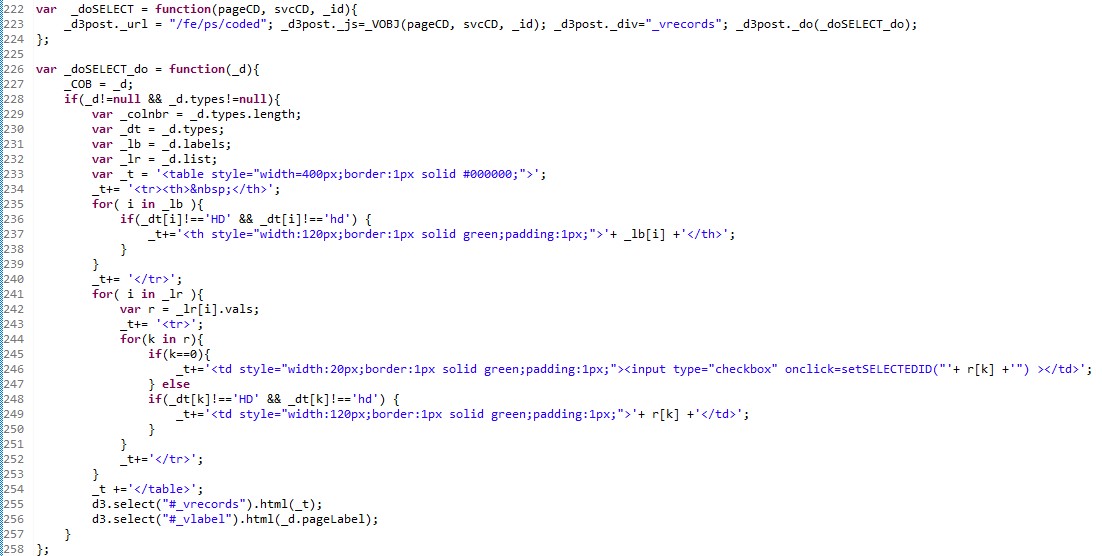


**File: quadrant\_api/main/com/vo/RecordDBVO.java**



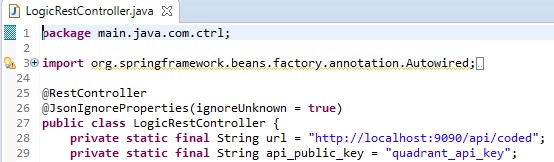
1. **Flow Read (SELECT)**
2. **Retrieve Service Command Code “SVCSEL” melalui fungsi js \_onSELECT() dari client side ke server side setelah page load ataupun salah satu table dipilih untuk di populate datanya ke view.**

**File: quadrant\_api\_caller/main/resource/public/landing.html**

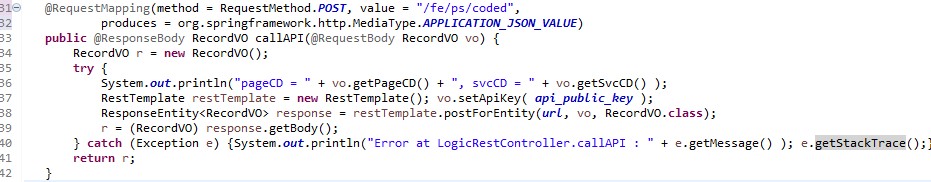


1. **Membuat POST request dari response SELECT tersebut di proses client side menggunakan RestTemplate**

**File: quadrant\_api\_caller/main/java/com/ctrl/LogicRestController.java**

****

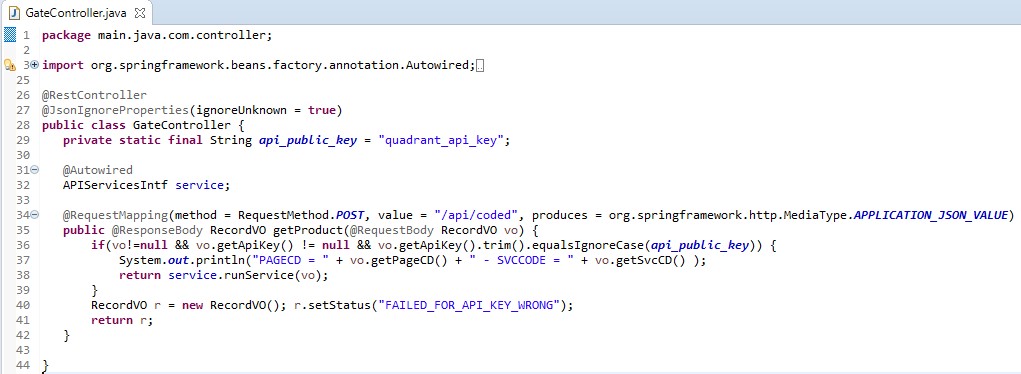
Client akan mengakses resources dari end point API dari server side dengan endpoint URL [***http://localhost:9090/api/coded***](http://localhost:9090/api/coded) menggunakan API key: ***quadrant\_api\_key***

****

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.

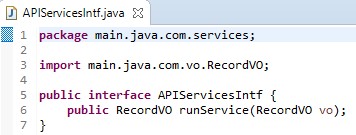
1. **Mereturn response HTTP setelah melakukan SELECT yang diproses server side melalui API Service yang masuk ke GateController**

**File: quadrant\_api/main/java/com/controller/GateController.java**

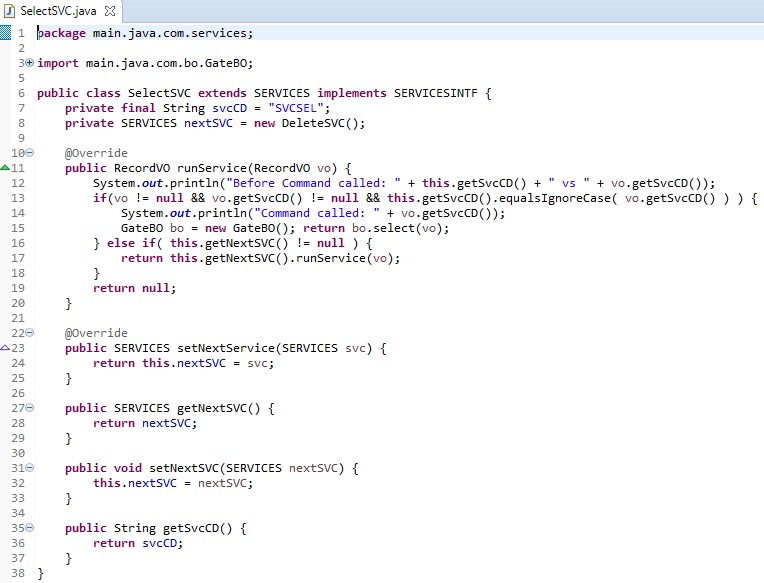
****

GateController 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**

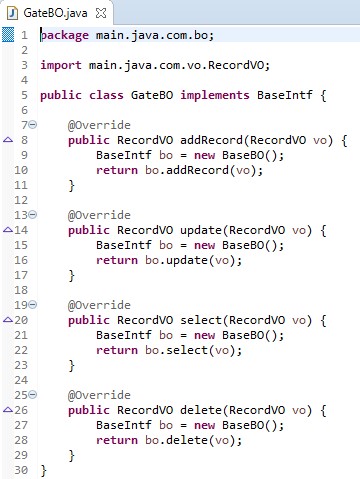
****

**File: quadrant\_api/main/java/com/services/SelectSVC.java**

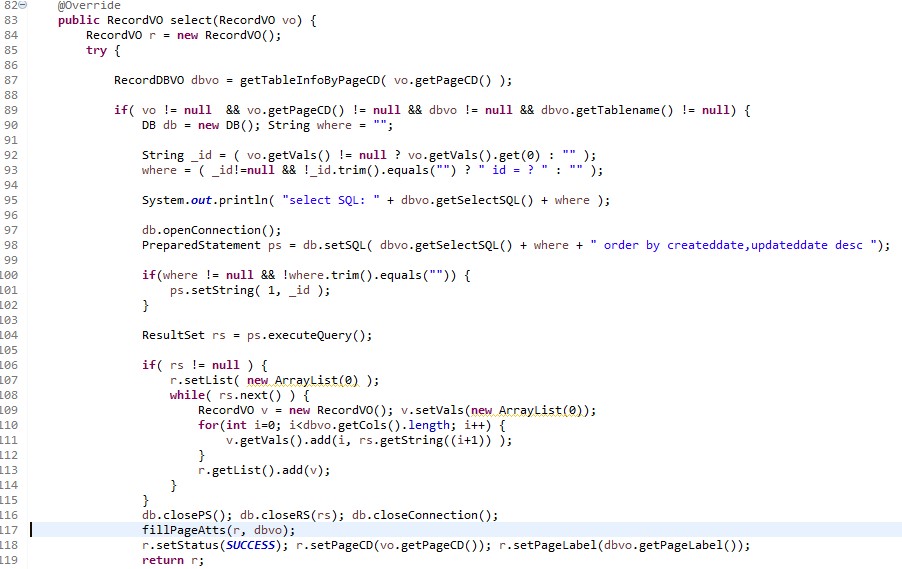


Dari service Select diatas request yang masuk akan di proses oleh Beans datasource transaction yakni **GateBO** yang akan menghandle jenis request apa yang akan di eksekusi query SQL nya. Disini SQL query yang akan di eksekusi adalah query select dengan memanggil method ***select()*** yang selanjutnya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

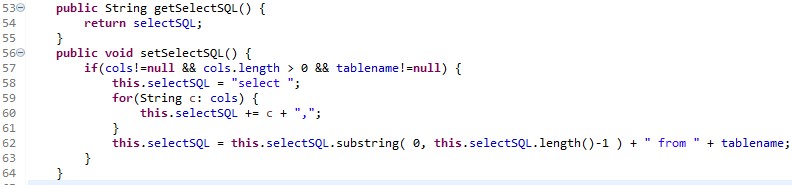
**File: quadrant\_api/main/com/bo/GateBO.java**

****

**File: quadrant\_api/main/com/bo/BaseBO .java**



**File: quadrant\_api/main/com/vo/RecordDBVO.java**



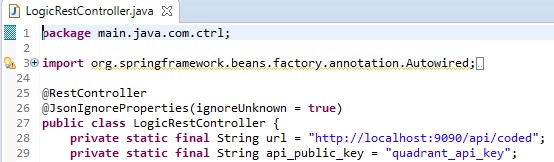
1. **Flow Update (UPDATE)**
2. **Retrieve Service Command Code “SVCUPD” melalui fungsi js \_onEDIT() dari client side ke server side setelah klik tombol EDIT.**

**File: quadrant\_api\_caller/main/resources/public/landing.html**

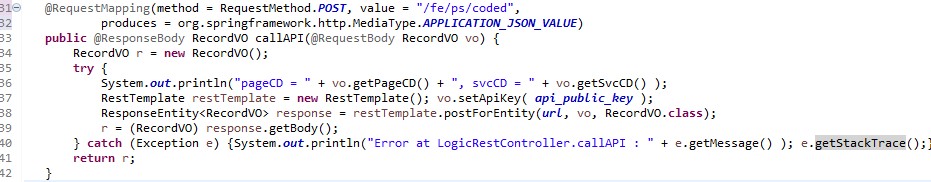


1. **Membuat POST request dari response UPDATE tersebut di proses client side menggunakan RestTemplate**

**File: quadrant\_api\_caller/main/java/com/ctrl/LogicRestController.java**

****

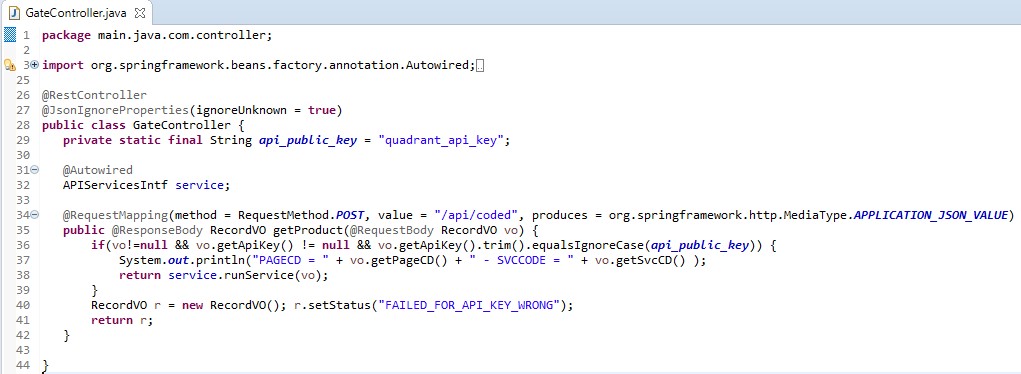
Client akan mengakses resources dari end point API dari server side dengan endpoint URL [***http://localhost:9090/api/coded***](http://localhost:9090/api/coded) menggunakan API key: ***quadrant\_api\_key***

****

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.

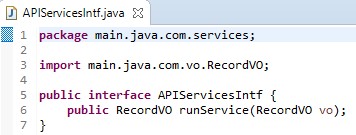
1. **Mereturn response HTTP setelah melakukan UPDATE yang diproses server side melalui API Service yang masuk ke GateController**

**File: quadrant\_api/main/java/com/controller/GateController.java**

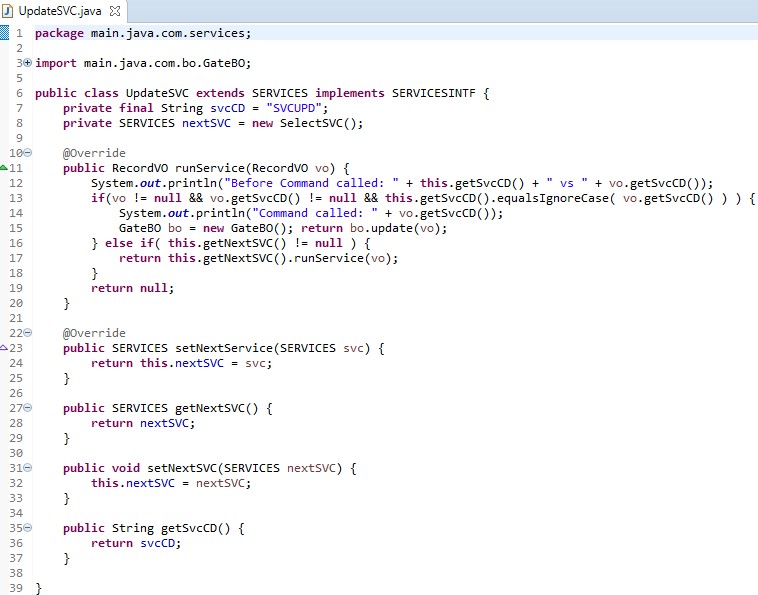
****

GateController 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**

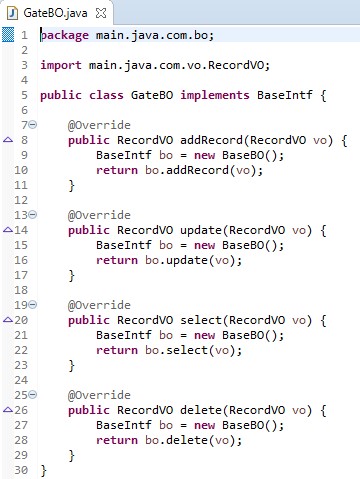
****

**File: quadrant\_api/main/java/com/services/UpdateSVC.java**

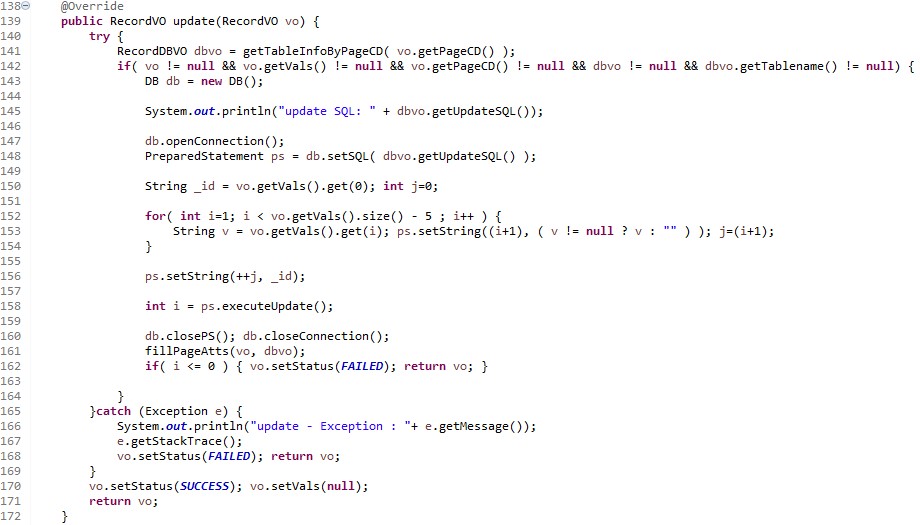


Dari service Update diatas request yang masuk akan di proses oleh Beans datasource transaction yakni **GateBO** yang akan menghandle jenis request apa yang akan di eksekusi query SQL nya. Disini SQL query yang akan di eksekusi adalah query update dengan memanggil method ***update()*** yang selanjutnya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

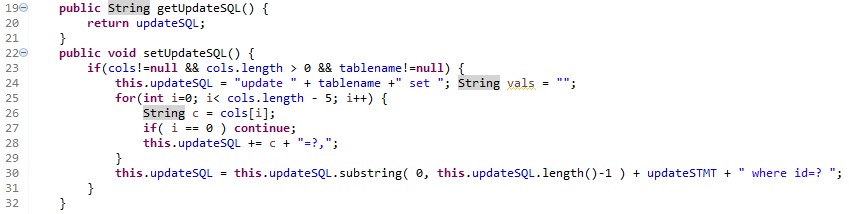
**File: quadrant\_api/main/com/bo/GateBO.java**

****

**File: quadrant\_api/main/com/bo/BaseBO .java**



**File: quadrant\_api/main/com/vo/RecordDBVO.java**



1. **Flow Delete (DELETE)**
2. **Retrieve Service Command Code “SVCDEL” melalui fungsi js \_onDELETE() dari client side ke server side setelah klik tombol DELETE.**

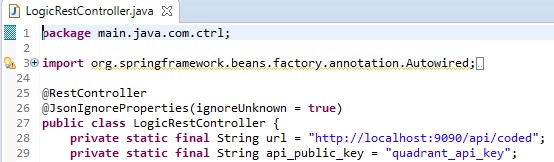
**File: quadrant\_api\_caller/main/resources/landing.html**



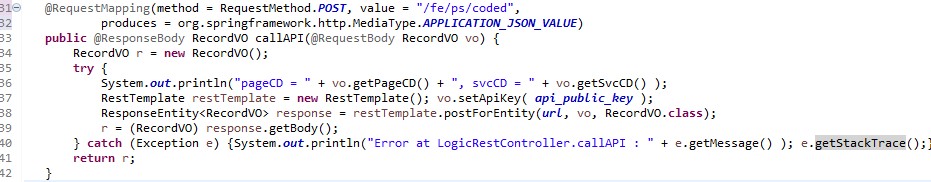
***\* fungsi onDelete pada file originallnya belum ada codenya saya coba tambahkan sendiri dengan mengikuti dari fungsi onEDIT karena sama-sama menselect Id.***

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

**File: quadrant\_api\_caller/main/java/com/ctrl/LogicRestController.java**

****

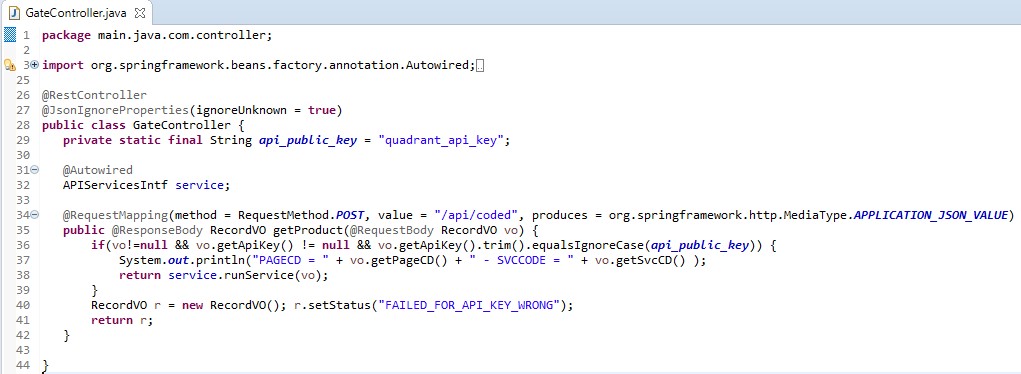
Client akan mengakses resources dari end point API dari server side dengan endpoint URL [***http://localhost:9090/api/coded***](http://localhost:9090/api/coded) menggunakan API key: ***quadrant\_api\_key***

****

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.

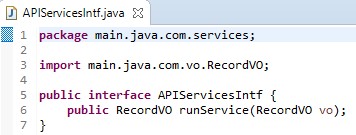
1. **Mereturn response HTTP setelah melakukan DELETE yang diproses server side melalui API Service yang masuk ke GateController**

**File: quadrant\_api/main/java/com/controller/GateController.java**

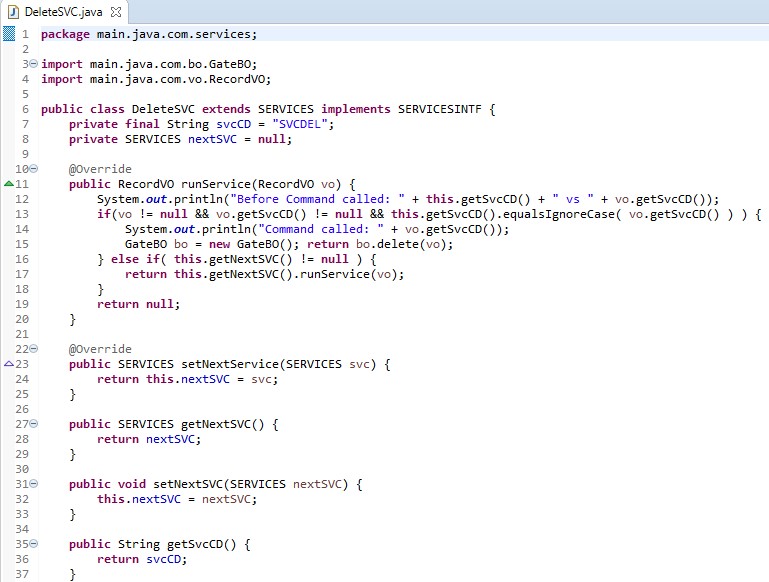
****

GateController 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**

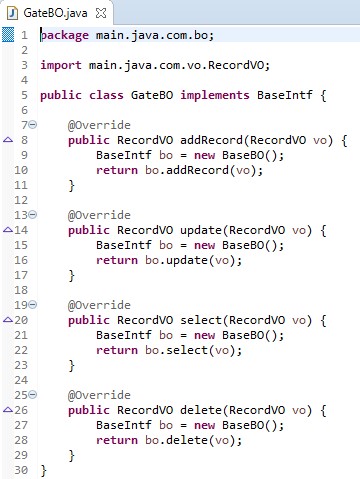
****

**File: quadrant\_api/main/java/com/services/DeleteSVC.java**



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**



**File: quadrant\_api/main/com/bo/BaseBO .java**



**File: quadrant\_api/main/com/vo/RecordDBVO.java**

