

Nama: Iman Alfathan Yudhanto

NPM: 1406623524

APAP-A

Write-up tutorial 7

Dalam pengerjaan latihan 1 dan 2. Penulis membuat folder untuk producer terlebih dahulu. Fungsi dari producer adalah untuk melakukan pengambilan data dan mempunyai *web service* untuk mengakses data-data StudentModel. Data bisa diambil langsung dari database ataupun sumber lain.

Latihan 1 dan latihan2 dikerjakan pada bagian Producer.

1. **Latihan 1:** Buatlah *service* untuk mengembalikan seluruh student yang ada di basis data. *Service* ini mirip seperti *method* viewAll di Web Controller. *Service* tersebut di-mapping ke `"/rest/student/viewall"`. Contoh tampilan keluarannya:

Untuk mengerjakan latihan 1, yang penulis lakukan pertama kali adalah membuat student rest controller. Fungsi dari student rest controller adalah mengembalikan objek yang ingin dikembalikan pada service. Spring Boot secara otomatis akan mengembalikan *output* berupa objek dengan format JSON pada tampilan Web. Data yang ditampilkan dapat berupa objek atau struktur data seperti Map, List, Stack, dll.

```
1 package com.example.rest;
2
3 import java.util.List;
4
5
6
7
8
9
10
11
12
13
14
15
16
17 @RestController
18 @RequestMapping("/rest")
19 public class StudentRestController {
20     @Autowired
21     StudentService studentService;
22
23     @RequestMapping("/student/view/{npm}")
24     public StudentModel view(@PathVariable(value = "npm") String npm) {
25         StudentModel student = studentService.selectStudent(npm);
26         return student;
27     }
28
29     @RequestMapping("/student/viewall")
30     public List<StudentModel> viewallRest(Model model) {
31         List<StudentModel> students = studentService.selectAllStudents();
32         return students;
33     }
34
35 }
```

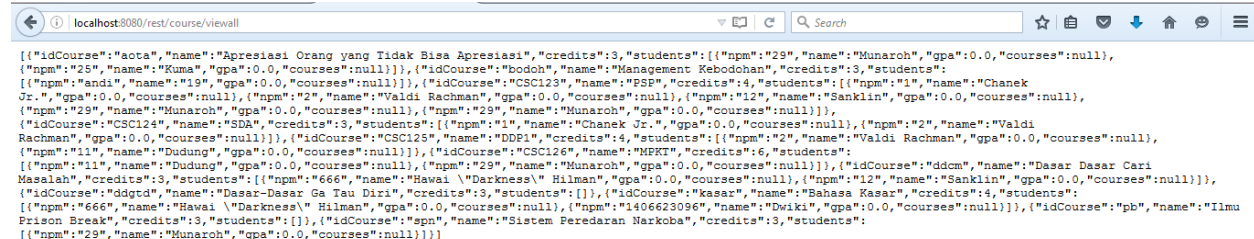
Untuk membuat *controller* menggunakan REST *web service* pada *class header* perlu diberikan anotasi **@RestController**, bukan @Controller seperti *controller* biasa. Ketika ingin mengaksesnya, maka perlu menuliskan <http://localhost:8080/rest/student/viewall>. Maka data yang ditampilkan adalah data berupa json. Seperti tampilan gambar di bawah:


```

1 package com.example.tutorial7_2_consumer/src/main/java/com/example/controller/CourseController.java
2
3 import java.util.List;
4
5 @RestController
6 @RequestMapping("/rest")
7 public class CourseRestController {
8
9     @Autowired
10    CourseService courseService;
11
12    @RequestMapping("/course/view/{idCourse}")
13    public CourseModel viewRest(@PathVariable(value = "idCourse") String idCourse) {
14        CourseModel course = courseService.selectCourse(idCourse);
15        return course;
16    }
17
18    @RequestMapping("/course/viewall")
19    public List<CourseModel> viewallRest(Model model) {
20        List<CourseModel> courses = courseService.selectAllCourse();
21        return courses;
22    }
23 }

```

Untuk membuat *controller* menggunakan REST *web service* pada *class header* perlu diberikan anotasi **@RestController**, bukan **@Controller** seperti *controller* biasa. Ketika ingin mengaksesnya, maka perlu menuliskan <http://localhost:8080/rest/course/viewall>. Maka data yang ditampilkan adalah data berupa json. Seperti tampilan gambar di bawah:

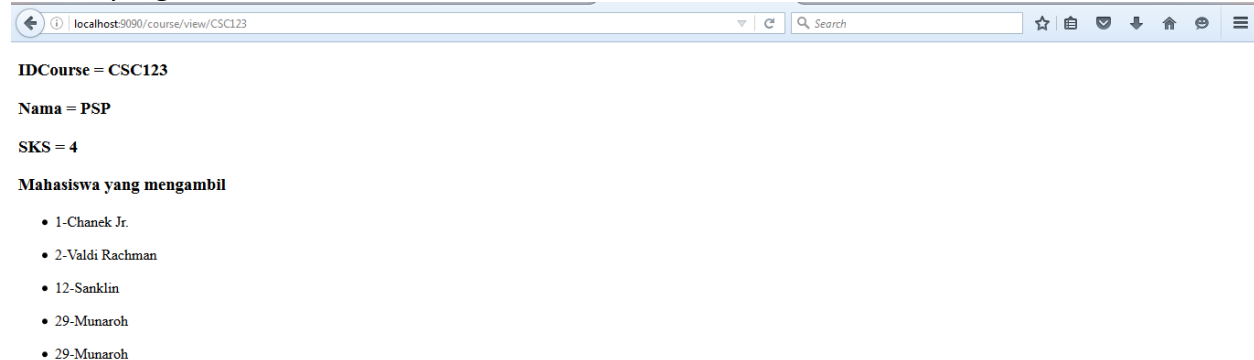


```

[{"idCourse": "aota", "name": "Apresiasi Orang yang Tidak Bisa Apresiasi", "credits": 3, "students": [{"npm": "29", "name": "Munaroh", "gpa": 0.0, "courses": null}, {"npm": "25", "name": "Kuma", "gpa": 0.0, "courses": null}], {"idCourse": "bodoh", "name": "Management Kebodohan", "credits": 3, "students": [{"npm": "19", "name": "PSP", "gpa": 0.0, "courses": null}], {"idCourse": "CSC123", "name": "PSP", "credits": 4, "students": [{"npm": "1", "name": "Chanek Jr.", "gpa": 0.0, "courses": null}, {"npm": "2", "name": "Valdi Rachman", "gpa": 0.0, "courses": null}, {"npm": "12", "name": "Sanklin", "gpa": 0.0, "courses": null}, {"npm": "29", "name": "Munaroh", "gpa": 0.0, "courses": null}], {"idCourse": "CSC124", "name": "SDA", "credits": 3, "students": [{"npm": "1", "name": "Chanek Jr.", "gpa": 0.0, "courses": null}, {"npm": "2", "name": "Valdi Rachman", "gpa": 0.0, "courses": null}], {"idCourse": "CSC125", "name": "DDP1", "credits": 4, "students": [{"npm": "2", "name": "Valdi Rachman", "gpa": 0.0, "courses": null}, {"npm": "11", "name": "Dudung", "gpa": 0.0, "courses": null}], {"idCourse": "CSC126", "name": "MPKT", "credits": 6, "students": [{"npm": "11", "name": "Dudung", "gpa": 0.0, "courses": null}, {"npm": "29", "name": "Munaroh", "gpa": 0.0, "courses": null}], {"idCourse": "ddcm", "name": "Dasar Dasar Cari Masalah", "credits": 3, "students": [{"npm": "666", "name": "Hawai \Darkness\ Hilman", "gpa": 0.0, "courses": null}, {"npm": "12", "name": "Sanklin", "gpa": 0.0, "courses": null}], {"idCourse": "ddgcd", "name": "Dasar-Dasar Ga Tau Diri", "credits": 3, "students": [{"npm": "1406623096", "name": "Dwiki", "gpa": 0.0, "courses": null}, {"npm": "666", "name": "Hawai \Darkness\ Hilman", "gpa": 0.0, "courses": null}], {"idCourse": "spn", "name": "Sistem Peredaran Narkoba", "credits": 3, "students": [{"npm": "29", "name": "Munaroh", "gpa": 0.0, "courses": null}]}

```

Hasil dari rest controller berupa objek-objek yang direpresentasikan dalam format JSON. Pada gambar di atas, data yang ditampilkan adalah idCourse, nama Course, dan credit milik semua Course yang ada.



```

{"idCourse": "CSC123", "name": "PSP", "credits": 4, "students": [{"npm": "1", "name": "Chanek Jr.", "gpa": 0.0, "courses": null}, {"npm": "2", "name": "Valdi Rachman", "gpa": 0.0, "courses": null}, {"npm": "12", "name": "Sanklin", "gpa": 0.0, "courses": null}, {"npm": "29", "name": "Munaroh", "gpa": 0.0, "courses": null}]}

```

IDCourse = CSC123
Nama = PSP
SKS = 4
Mahasiswa yang mengambil

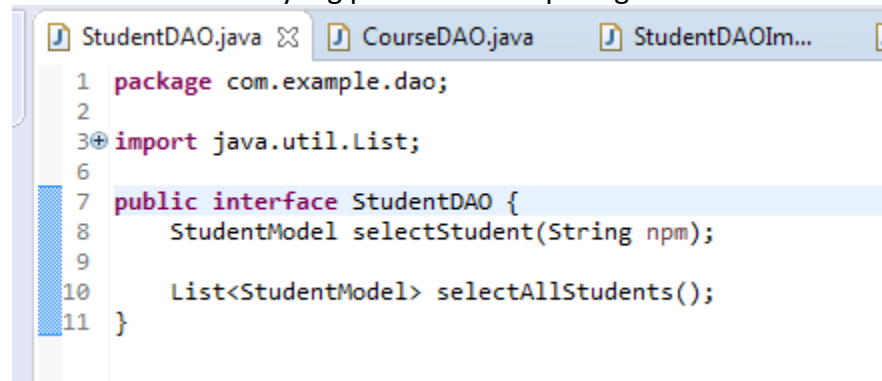
- 1-Chanek Jr.
- 2-Valdi Rachman
- 12-Sanklin
- 29-Munaroh

Gambar di atas adalah tampilan jika berhasil mengakses <http://localhost:9090/course/view/{idCourse}>. Data yang ditampilkan adalah idCourse, nama Course, dan credit milik Course berdasarkan npm yang dicari.

Dalam pengerjaan latihan 3 dan 4. Penulis membuat folder untuk consumer terlebih dahulu. *Service Consumer* berfungsi untuk mengonsumsi *web service* untuk mengakses data-data *StudentModel* dari bagian *Producer*. Pada bagian *consumer*, penulis tidak menggunakan mapper seperti tutorial yang sebelumnya karena database telah dihubungkan oleh producer. Latihan 3 dan latihan 4 dikerjakan pada bagian *Consumer*.

1. Latihan 3: Implementasikan *service consumer* untuk view all Students dengan melengkapi *method* *selectAllStudents* yang ada di kelas *StudentServiceRest*.

Untuk mengerjakan latihan 1, yang penulis lakukan pertama kali adalah membuat *StudentDAO*. *StudentDAO* memiliki fungsi yang hampir sama seperti *StudentService* di tutorial sebelumnya. Contoh *studentDAO* yang penulis buat seperti gambar di bawah.



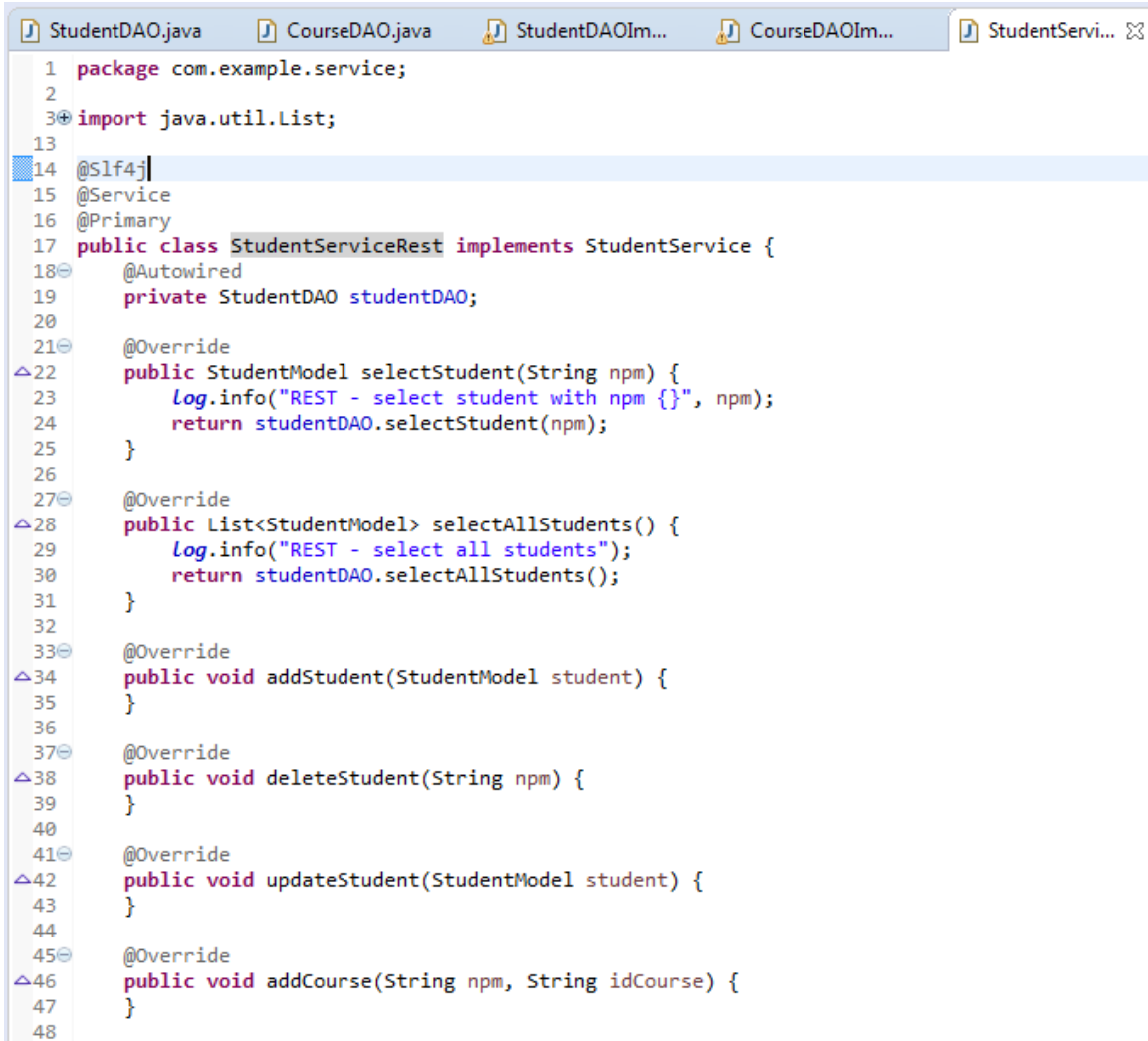
```
1 package com.example.dao;
2
3 import java.util.List;
4
5
6
7 public interface StudentDAO {
8     StudentModel selectStudent(String npm);
9
10    List<StudentModel> selectAllStudents();
11 }
```

Di *StudentDAO*, penulis membuat interface yang berisi method seperti gambar di atas. Setelah itu penulis membuat *StudentDAOImpl*.



```
1 package com.example.dao;
2
3 import java.util.List;
4
5
6
7 @Service
8 public class StudentDAOImpl implements StudentDAO {
9     @Autowired
10    private RestTemplate restTemplate;
11
12    @Override
13    public StudentModel selectStudent(String npm) {
14        StudentModel student = restTemplate.getForObject("http://localhost:8080/rest/student/view/" + npm,
15            StudentModel.class);
16        return student;
17    }
18
19    @Override
20    public List<StudentModel> selectAllStudents() {
21        List<StudentModel> student = restTemplate.getForObject("http://localhost:8080/rest/student/viewall",
22            List.class);
23        return student;
24    }
25 }
```

Gambar di atas merupakan StudentDAOImpl yang dibuat oleh penulis. Kelas tersebut mengimplementasi interface StudentDAO. Pada kelas tersebut, terdapat sebuah method bernama getObject. Gunanya adalah memanggil controller milik producer sehingga consumer bisa mengakses data yang diambil oleh producer. Kelas tersebut meng-autowired RestTemplate agar bisa mengakses data dari producer. Setelah itu, penulis membuat StudentServiceRest.



```
1 package com.example.service;
2
3 import java.util.List;
4
5 @Slf4j
6 @Service
7 @Primary
8 public class StudentServiceRest implements StudentService {
9     @Autowired
10     private StudentDAO studentDAO;
11
12     @Override
13     public StudentModel selectStudent(String npm) {
14         log.info("REST - select student with npm {}", npm);
15         return studentDAO.selectStudent(npm);
16     }
17
18     @Override
19     public List<StudentModel> selectAllStudents() {
20         log.info("REST - select all students");
21         return studentDAO.selectAllStudents();
22     }
23
24     @Override
25     public void addStudent(StudentModel student) {
26     }
27
28     @Override
29     public void deleteStudent(String npm) {
30     }
31
32     @Override
33     public void updateStudent(StudentModel student) {
34     }
35
36     @Override
37     public void addCourse(String npm, String idCourse) {
38     }
39 }
```

Gambar di atas adalah StudentServiceRest yang dibuat oleh penulis. Kelas tersebut mengimplementasi interface StudentService. Guna dari kelas tersebut adalah mengakses interface DAO, sehingga bisa memanggil data yang diakses oleh producer. Yang membedakan kelas service ini dengan kelas service pada tutorial sebelumnya adalah kelas ini mengakses DAO. Sedangkan kelas service di tutorial sebelumnya mengakses mapper. Agar studentController menggunakan ServiceRest yang telah dibuat, maka digunakan anotasi **@Primary**. Autowired akan secara otomatis menginstansiasi StudentService menggunakan StudentServiceRest karena dia *primary*.

```

45     return "success-add";
46 }
47
48 @RequestMapping("/student/view")
49 public String view(Model model, @RequestParam(value = "npm", required = false) String npm) {
50     StudentModel student = studentService.selectStudent(npm);
51
52     if (student != null) {
53         model.addAttribute("student", student);
54         return "view";
55     } else {
56         model.addAttribute("npm", npm);
57         return "not-found";
58     }
59 }
60
61 @RequestMapping("/student/view/{npm}")
62 public String viewPath(Model model, @PathVariable(value = "npm") String npm) {
63     StudentModel student = studentService.selectStudent(npm);
64
65     if (student != null) {
66         model.addAttribute("student", student);
67         return "view";
68     } else {
69         model.addAttribute("npm", npm);
70         return "not-found";
71     }
72 }
73
74 @RequestMapping("/student/viewall")
75 public String view(Model model) {
76     List<StudentModel> students = studentService.selectAllStudents();
77     model.addAttribute("students", students);
78
79     return "viewall";
80 }

```

Gambar di atas adalah StudentController yang penulis buat. Tidak ada yang dirubah pada kelas tersebut. Fungsi pada gambar di atas menampilkan fungsi untuk melakukan viewall dan view detail. Ketika ingin mengakses viewall student, maka perlu menuliskan <http://localhost:9090/student/viewall>. Maka data yang ditampilkan adalah seperti tampilan gambar di bawah:

localhost:9090/student/viewall

View All Mahasiswa

Show 10 entries

Search:

No	NPM	Nama	GPA	Status Kelulusan	Matkul yang diambil	Update	Delete
1	1	Chanek Jr.	3.41	Sangat Memuaskan	<ul style="list-style-type: none"> PSP-4 sks SDA-3 sks 	<button>Update</button>	<button>Delete</button>
2	11	Dudung	3.0	Sangat Memuaskan	<ul style="list-style-type: none"> MPKT-6 sks DDP1-4 sks 	<button>Update</button>	<button>Delete</button>
3	12	Sanklin	3.11	Sangat Memuaskan	<ul style="list-style-type: none"> Dasar Dasar Cari Masalah-3 sks PSP-4 sks 	<button>Update</button>	<button>Delete</button>
4	13	sabodo teuing	4.3	Cum Laude!		<button>Update</button>	<button>Delete</button>
5	14	gikbasembrani@gmail.com	3.0	Sangat Memuaskan		<button>Update</button>	<button>Delete</button>
6	1406623096	Dwiki	-2.3	Sangat Memuaskan	<ul style="list-style-type: none"> Bahasa Kasar-4 sks 	<button>Update</button>	<button>Delete</button>
7	15	Luffy	3.33	Sangat Memuaskan		<button>Update</button>	<button>Delete</button>
8	2	Valdi Rachman	4.0	Cum Laude!	<ul style="list-style-type: none"> PSP-4 sks 	<button>Update</button>	<button>Delete</button>

Jika ingin mengakses view detail student, maka perlu menuliskan <http://localhost:9090/student/view/{npm}>. Maka data yang ditampilkan adalah seperti tampilan gambar di bawah:

NPM = 1

Name = Chanek Jr.

GPA = 3.41

Kuliah yang diambil

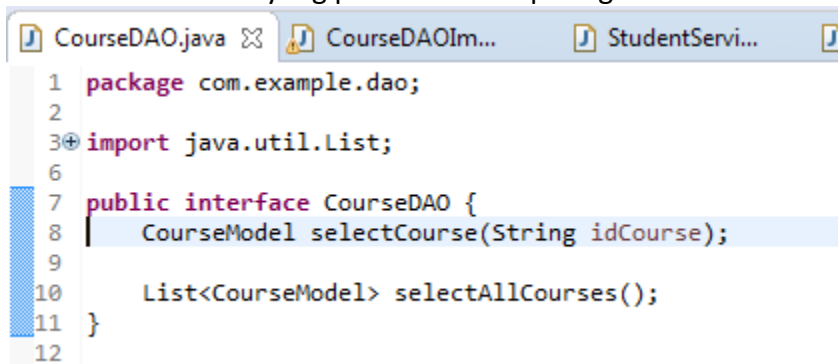
- PSP-4 sks
- SDA-3 sks

Jika consumer sudah bisa terakses melalui `studentServiceRest`, maka akan muncul penjelasan pada console.

```
2017-11-03 22:02:13.802 INFO 3040 --- [nio-9090-exec-1] com.example.service.StudentServiceRest : REST - select all students
2017-11-03 22:06:25.720 INFO 3040 --- [nio-9090-exec-4] com.example.service.StudentServiceRest : REST - select student with npm 1
```


2. Latihan 4: Implementasikan *service consumer* untuk *class* `CourseModel` dengan membuat *class-class* DAO dan *service* baru.

Untuk mengerjakan latihan 1, yang penulis lakukan pertama kali adalah membuat `CourseDAO`. `CourseDAO` memiliki fungsi yang hampir sama seperti `CourseService` di tutorial sebelumnya. Contoh `CourseDAO` yang penulis buat seperti gambar di bawah.



```
1 package com.example.dao;
2
3 import java.util.List;
4
5
6
7 public interface CourseDAO {
8     CourseModel selectCourse(String idCourse);
9
10    List<CourseModel> selectAllCourses();
11 }
12
```

Di `CourseDAO`, penulis membuat interface yang berisi mehod seperti gambar di atas. Setelah itu penulis membuat `CourseDAOImpl`.

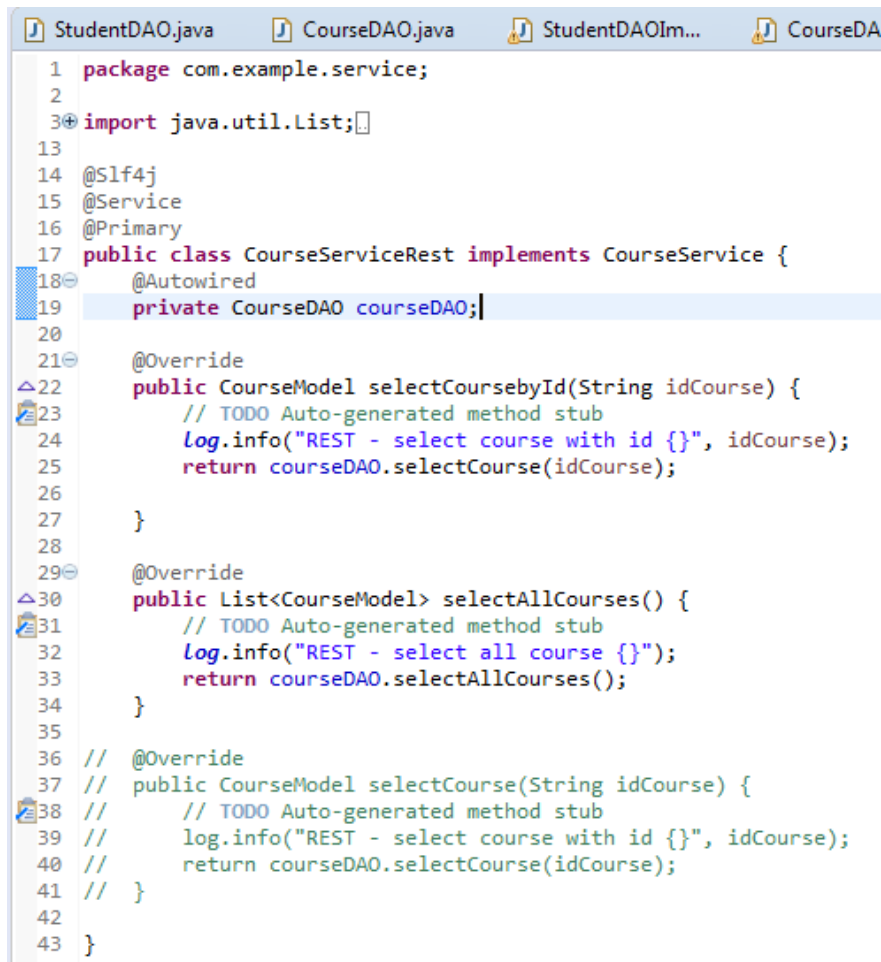


```

1 package com.example.dao;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.boot.web.client.RestTemplateBuilder;
7 import org.springframework.context.annotation.Bean;
8 import org.springframework.stereotype.Service;
9 import org.springframework.web.client.RestTemplate;
10
11 import com.example.model.CourseModel;
12
13 @Service
14 public class CourseDAOImpl implements CourseDAO {
15     @Autowired
16     private RestTemplate restTemplate;
17
18     @Override
19     public CourseModel selectCourse(String idCourse) {
20         // TODO Auto-generated method stub
21         CourseModel course = restTemplate.getForObject("http://localhost:8080/rest/course/view/" + idCourse,
22             CourseModel.class);
23         return course;
24     }
25
26     @Override
27     public List<CourseModel> selectAllCourses() {
28         // TODO Auto-generated method stub
29         List<CourseModel> courses = restTemplate.getForObject("http://localhost:8080/rest/course/viewall",
30             List.class);
31         return courses;
32     }
33
34     @Bean
35     public RestTemplate restTemplate(RestTemplateBuilder builder) {
36         // Do any additional configuration here
37         return builder.build();
38     }

```

Gambar di atas merupakan CourseDAOImpl yang dibuat oleh penulis. Kelas tersebut mengimplementasi interface CourseDAO. Pada kelas tersebut, terdapat sebuah method bernama getForObject. Gunanya adalah memanggil controller milik producer sehingga consumer bisa mengakses data yang diambil oleh producer. Kelas tersebut meng-autowired RestTemplate agar bisa mengakses data dari producer. Setelah itu, penulis membuat CourseServiceRest.



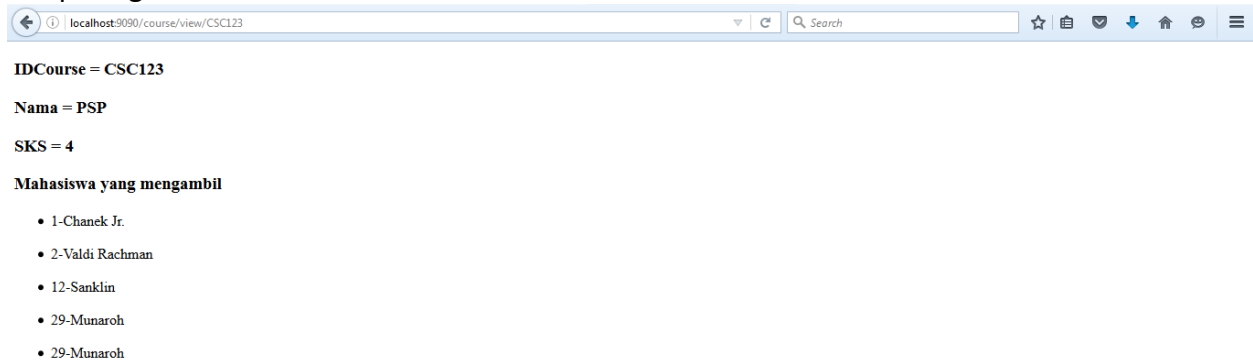
```

1  package com.example.service;
2
3  import java.util.List;
4
13
14  @Slf4j
15  @Service
16  @Primary
17  public class CourseServiceRest implements CourseService {
18      @Autowired
19      private CourseDAO courseDAO;
20
21      @Override
22      public CourseModel selectCourseById(String idCourse) {
23          // TODO Auto-generated method stub
24          log.info("REST - select course with id {}", idCourse);
25          return courseDAO.selectCourse(idCourse);
26      }
27
28
29      @Override
30      public List<CourseModel> selectAllCourses() {
31          // TODO Auto-generated method stub
32          log.info("REST - select all course {}");
33          return courseDAO.selectAllCourses();
34      }
35
36      // @Override
37      // public CourseModel selectCourse(String idCourse) {
38      //     // TODO Auto-generated method stub
39      //     log.info("REST - select course with id {}", idCourse);
40      //     return courseDAO.selectCourse(idCourse);
41      // }
42
43  }

```

Gambar di atas adalah CourseServiceRest yang dibuat oleh penulis. Kelas tersebut mengimplement interface CourseService. Guna dari kelas tersebut adalah mengakses interface DAO, sehingga bisa memanggil data yang diakses oleh producer. Yang membedakan kelas service ini dengan kelas service pada tutorial sebelumnya adalah kelas ini mengakses DAO. Sedangkan kelas service di tutorial sebelumnya mengakses mapper. Agar CourseController menggunakan ServiceRest yang telah dibuat, maka digunakan anotasi **@Primary**. *Autowired* akan secara otomatis menginstansiasi CourseService menggunakan CourseServiceRest karena dia *primary*. Tidak ada yang dirubah pada kelas CourseController.

Jika ingin mengakses view detail course, maka perlu menuliskan <http://localhost:9090/course/view/{idCourse}>. Maka data yang ditampilkan adalah seperti tampilan gambar di bawah:



Jika ingin mengakses viewall course, maka perlu menuliskan <http://localhost:9090/course/viewall>. Maka data yang ditampilkan adalah seperti tampilan gambar di bawah:

The screenshot shows a web browser window with the address bar displaying `localhost:9090/course/viewall`. The page title is "View all Course". Below the title, there is a "Show 10 entries" dropdown and a "Search:" input field. The main content is a table with the following data:

No	ID Course	Nama	Student yang mengambil
1	aota	Apresiasi Orang yang Tidak Bisa Apresiasi	<ul style="list-style-type: none">• 29-Munaroh• 25-Kuma
2	bodoh	Management Kebodohan	<ul style="list-style-type: none">• andi-19
3	CSC123	PSP	<ul style="list-style-type: none">• 1-Chanek Jr.• 2-Valdi Rachman• 12-Sanklin• 29-Munaroh• 29-Munaroh
4	CSC124	SDA	<ul style="list-style-type: none">• 1-Chanek Jr.• 2-Valdi Rachman
5	CSC125	DDP1	<ul style="list-style-type: none">• 2-Valdi Rachman• 11-Dudung
6	CSC126	MPKT	<ul style="list-style-type: none">• 11-Dudung

LESSON LEARNED:

- Menggunakan REST. Rest biasa digunakan pada bagian producer. Terdapat anotasi `@RestController` agar menjadikan controller tersebut menjadi controller dengan format Rest. Fungsinya menandakan bahwa ia adalah si producer, dapat me-return data berupa objek model, dan menggunakan web service.
- Mengambil data dari program yang berbeda.
- Pemanfaat REST dapat digunakan untuk memberikan data menuju sistem yang berbeda.
- Penggunaan REST dapat mengembalikan data dalam bentuk berupa JSON.

- Perlu adanya atribut RestTemplate untuk menembak ke sistem lain (menghubungkan ke producer).
- Sistem yang diambil datanya cukup menyediakan JSON saja, sedangkan sistem yang menggunakan data (*consumer*) cukup memiliki *controller*, DAO interface dan kelas yang berfungsi untuk berhubungan dengan producer), service untuk menghubungkan dengan DAO yang memadai, serta halaman html.
- Terdapat anotasi primary untuk menjadikan sebuah service menjadi service utama.