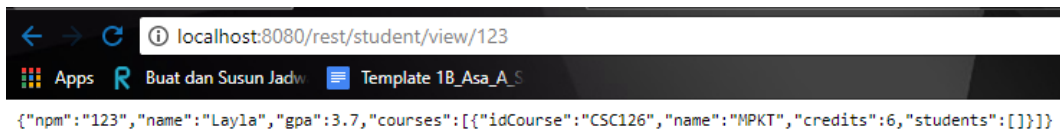


Membuat *Service Producer*

1. Buat *class* baru dengan nama `StudentRestController` dan coba jalankan `“localhost:8080/rest/student/view/124”`



Yang dikembalikan merupakan object `StudentModel` dalam format JSON.

Untuk memudahkan pembacaan silakan gunakan JSON View di link berikut: <http://json.parser.online.fr/>. Hasilnya adalah sebagai berikut:



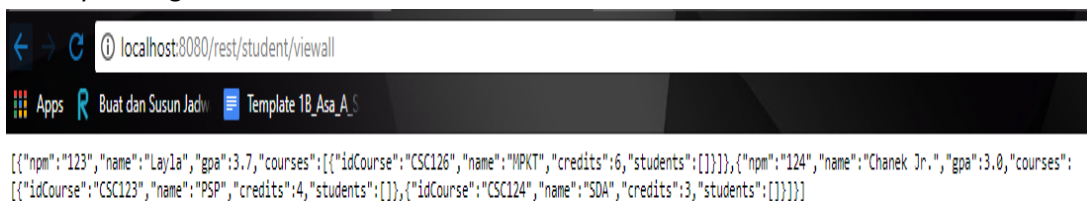
LATIHAN

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”`.

Mengembalikan seluruh *student* berarti kita menggunakan list dengan tipe *student model* yang akan mereturn *students*.

```
26 @RequestMapping("/student/viewall")
27 public List<StudentModel> viewall (StudentModel model) {
28     List<StudentModel> students = studentService.selectAllStudents ();
29     return students;
30 }
```

Hasilnya sebagai berikut:



```
String parse
[
  {
    "npm": "123",
    "name": "Layla",
    "gpa": 3.7,
    "courses": [
      {
        "idCourse": "CSCI26",
        "name": "MPKI",
        "credits": 6,
        "students": [
        ]
      }
    ]
  },
  {
    "npm": "124",
    "name": "Chanek Jr.",
    "gpa": 3.0,
    "courses": [
      {
        "idCourse": "CSCI23",
        "name": "PSP",
        "credits": 4,
        "students": [
        ]
      },
      {
        "idCourse": "CSCI24",
        "name": "SDA",
        "credits": 3,
        "students": [
        ]
      }
    ]
  }
]
}
```

2. Buatlah *service* untuk *class Course*. Buatlah *controller* baru yang terdapat *service* untuk melihat suatu *course* dengan masukan *ID Course (view by ID)* dan *service* untuk melihat semua *course (view all)*.

Jika latihan nomor 1 untuk melihat students, latihan nomer 2 untuk melihat course nya. Karena method view course by id sudah dibuat pada tutorial 5, tahap selanjutnya adalah membuat method view all course pada controller rest. Sebelumnya, harus membuat method viewall untuk course pada student controller, mapper dan html.

- Buat *method viewall* pada *class StudentController*

```
160 @RequestMapping("/course/viewall")
161 public String viewAllCourse (Model model)
162 {
163     List<CourseModel> courses = studentDAO.selectAllCourse();
164     model.addAttribute ("courses", courses);
165     return "viewall-course";
166 }
```

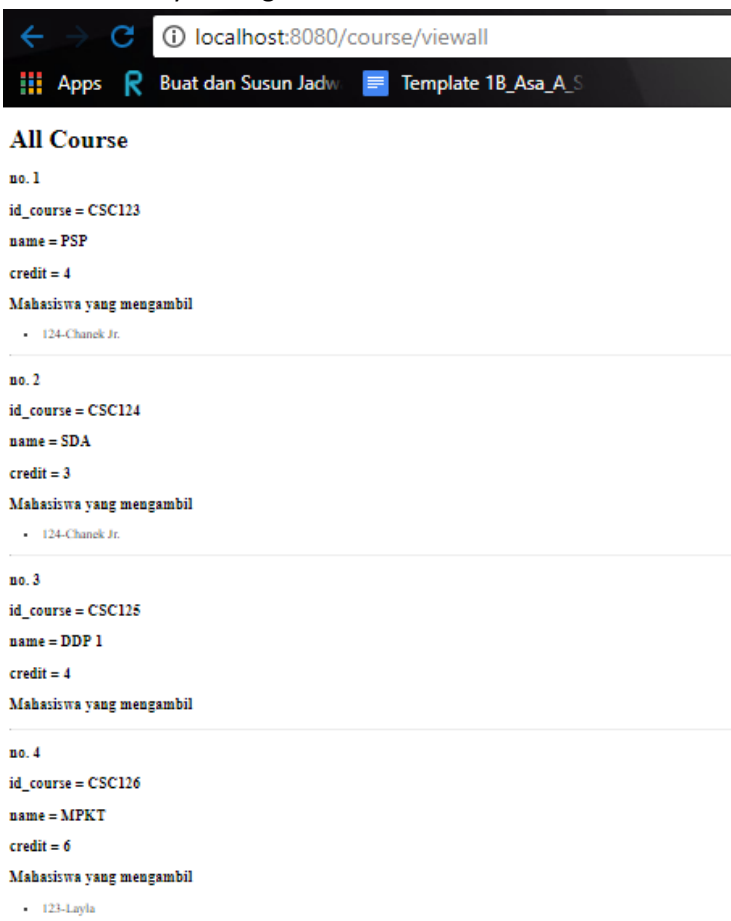
- Buat file html nya

```
viewall-course.html
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4 <title>View All Course</title>
5 </head>
6 <body>
7 <h1>All Course</h1>
8
9 <div th:each="course, iterationStatus: ${courses}">
10 <h3 th:text="'no. ' + ${iterationStatus.count}">No. 1</h3>
11 <h3 th:text="'id_course = ' + ${course.idCourse}">id course</h3>
12 <h3 th:text="'name = ' + ${course.name}">course Name</h3>
13 <h3 th:text="'credit = ' + ${course.credits}">credit</h3>
14
15 <h3>Mahasiswa yang mengambil</h3>
16 <ul th:each="students, iterationStatus: ${course.students}">
17 <li th:text="${students.npm} + '-' + ${students.name}">
18 </li>
19 </ul>
20 <hr />
21 </div>
22 </body>
23 </html>
```

- Buat Mapper untuk *viewall-course*

```
71 @Select("SELECT * from course")
72 @Results(value = {
73     @Result(property="idCourse", column="id_course"),
74     @Result(property="name", column="name"),
75     @Result(property="credits", column="credits"),
76     @Result(property="students", column="id_course", javaType = List.class, many = @Many(select="selectCourseStudent"))
77 })
78 List<CourseModel> selectAllCourse();
```

- Hasilnya sebagai berikut :



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/course/viewall'. The page content is as follows:

All Course

no. 1
id_course = CSC123
name = PSP
credit = 4
Mahasiswa yang mengambil
• 124-Chanek Jr.

no. 2
id_course = CSC124
name = SDA
credit = 3
Mahasiswa yang mengambil
• 124-Chanek Jr.

no. 3
id_course = CSC125
name = DDP 1
credit = 4
Mahasiswa yang mengambil

no. 4
id_course = CSC126
name = MPKT
credit = 6
Mahasiswa yang mengambil
• 123-Layla

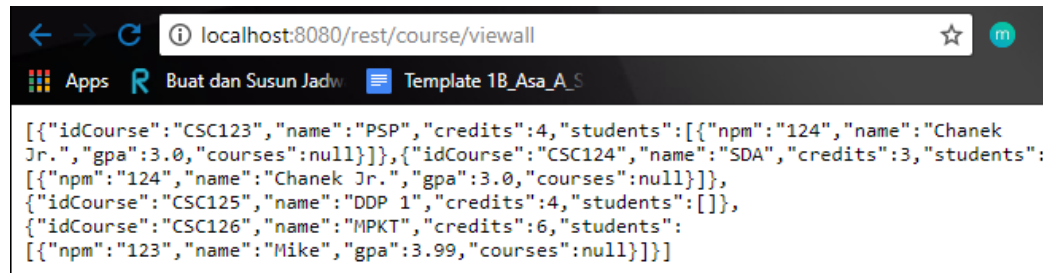
Lalu buat RequestMapping pada rest controller

```

26 @RequestMapping("/student/viewall")
27 public List<StudentModel> viewall (StudentModel model) {
28     List<StudentModel> students = studentService.selectAllStudents ();
29     return students;
30 }

```

Maka bila dijalankan localhost:8080/rest/course/viewall seperti berikut



```

[{"idCourse":"CSC123","name":"PSP","credits":4,"students":[{"npm":"124","name":"Chanek Jr.", "gpa":3.0,"courses":null}]}, {"idCourse":"CSC124","name":"SDA","credits":3,"students":[{"npm":"124","name":"Chanek Jr.", "gpa":3.0,"courses":null}]}, {"idCourse":"CSC125","name":"DDP 1","credits":4,"students":[]}, {"idCourse":"CSC126","name":"MPKT","credits":6,"students":[{"npm":"123","name":"Mike", "gpa":3.99,"courses":null}]}]

```

String parse

```

[{"idCourse":"CSC123","name":"PSP","credits":4,"students":[{"npm":"124","name":"Chanek Jr.", "gpa":3.0,"courses":null}]}, {"idCourse":"CSC124","name":"SDA","credits":3,"students":[{"npm":"124","name":"Chanek Jr.", "gpa":3.0,"courses":null}]}, {"idCourse":"CSC125","name":"DDP 1","credits":4,"students":[]}, {"idCourse":"CSC126","name":"MPKT","credits":6,"students":[{"npm":"123","name":"Mike", "gpa":3.99,"courses":null}]}]

```

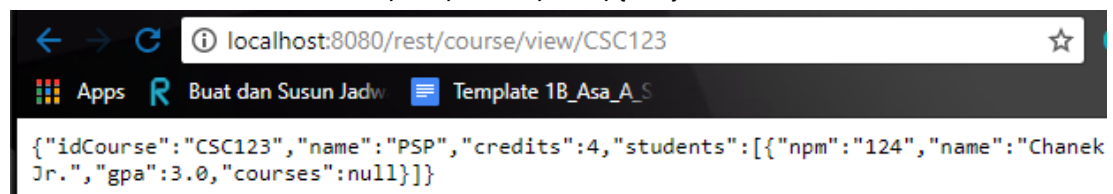
- Sekarang view course by id, tambahkan requestMapping berikut pada rest controller

```

39 @RequestMapping("/course/view/{id}")
40 public CourseModel viewCourse(@PathVariable(value = "id") String id_course) {
41     CourseModel course = studentService.viewCourse(id_course);
42     return course;
43 }

```

- Jalankan localhost:8080/rest/course/view/{123}



```

{"idCourse":"CSC123","name":"PSP","credits":4,"students":[{"npm":"124","name":"Chanek Jr.", "gpa":3.0,"courses":null}]}

```

```
String parse

{
  "idCourse": "CSC123", "name": "PSP", "credits": 4, "students": [
    {
      "npm": "124", "name": "Chanek Jr.", "gpa": 3.0, "courses": null
    }
  ]
}
```

Membuat *Service Consumer*

1. Jalankan kedua project Spring Boot untuk Service Producer dan Service Consumer tersebut.
2. Pastikan service producer sudah berjalan dengan menjalankan localhost:8080/rest/student/view/123.
3. Untuk menguji service consumer buka localhost:9090/student/view/123

```
localhost:9090/student/view/123

NPM = 123

Name = Mike

GPA = 3.99

Kuliah yang diambil
• MPKT-6 sks
```

4. Pada *console* tertulis:

```
[nio-9090-exec-1] com.example.service.StudentServiceRest : REST - select student with npm123
```

LATIHAN

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

Implementasi akan dilakukan di *class StudentDAOImpl.java* seperti berikut:

```
26 @Override
27 public List<StudentModel> selectAllStudents ()
28 {
29     List<StudentModel> student = restTemplate.getForObject("http://localhost:8080/rest/student/viewall", List.class);
30     return student;
31 }
```

Lalu pada *StudentServiceRest.java*

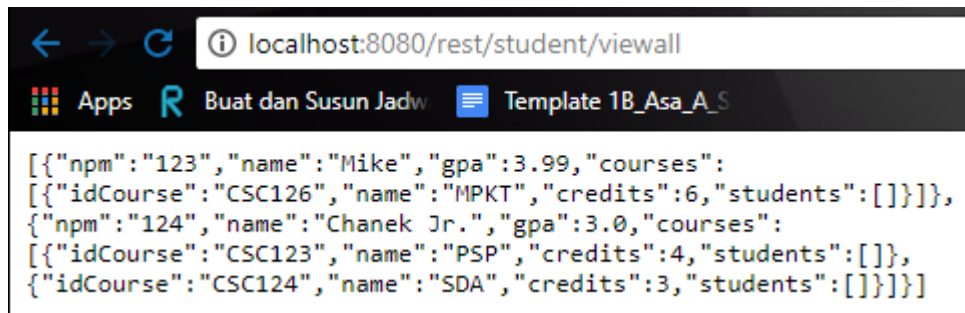
```

37 @Override
38 public List<StudentModel> selectAllStudents ()
39 {
40     //log.info ("REST - select all students");
41     return StudentDAO.selectAllStudents();
42 }

```

Sekarang, jalankan program, akses localhost:8080/rest/student/viewall

Hasilnya seperti berikut:



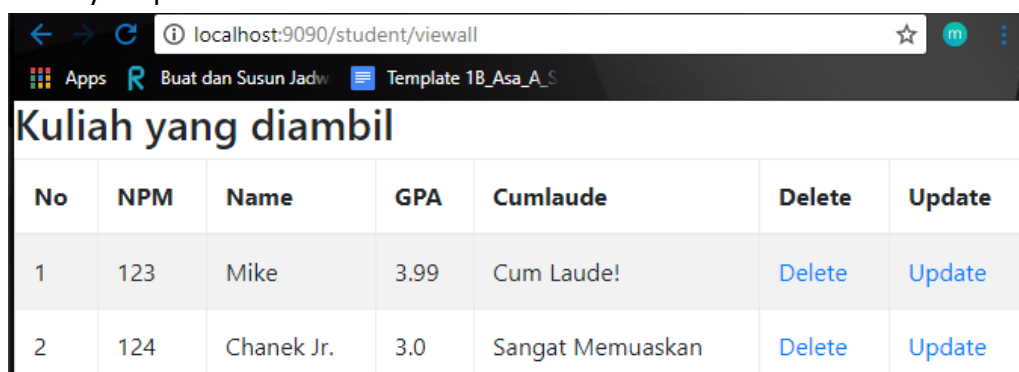
```

[{"npm":"123","name":"Mike","gpa":3.99,"courses":
[{"idCourse":"CSC126","name":"MPKT","credits":6,"students":[]}]},
{"npm":"124","name":"Chanek Jr.","gpa":3.0,"courses":
[{"idCourse":"CSC123","name":"PSP","credits":4,"students":[]},
{"idCourse":"CSC124","name":"SDA","credits":3,"students":[]}]}}]

```

Sekarang, jalankan program, akses localhost:9090/student/viewall

Hasilnya seperti berikut:



No	NPM	Name	GPA	Cumlaude	Delete	Update
1	123	Mike	3.99	Cum Laude!	Delete	Update
2	124	Chanek Jr.	3.0	Sangat Memuaskan	Delete	Update

- Implementasikan *service consumer* untuk *class CourseModel* dengan membuat *class-DAO* dan *service* baru.

- Buat *interface CourseDAO* pada *package dao*. Berisi seperti berikut :

```

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

```

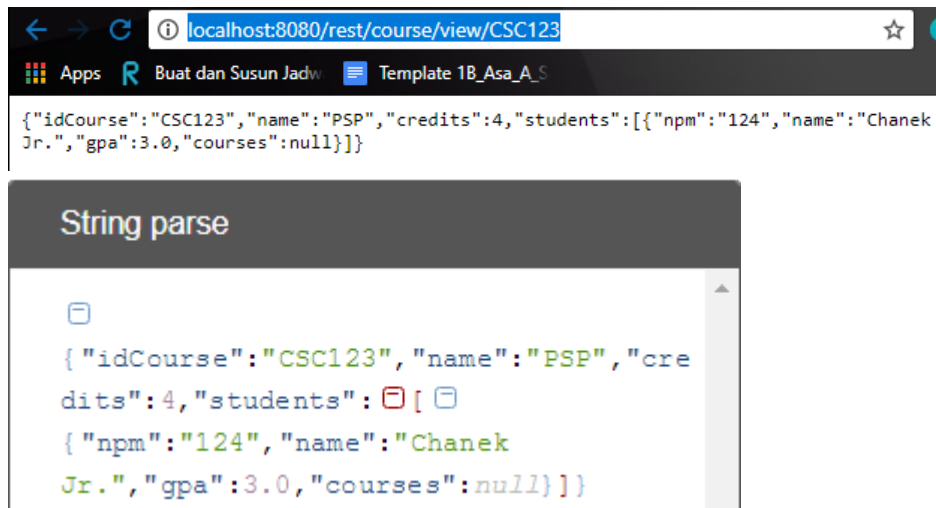
2. Implementasikan pada *class* CourseDAOImpl.java seperti berikut:

```

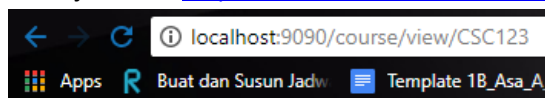
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7 import org.springframework.web.client.RestTemplate;
8
9 import com.example.model.CourseModel;
10
11 @Service
12 public class CourseDAOImpl implements CourseDAO{
13
14     @Autowired
15     private RestTemplate restTemplate;
16
17     @Override
18     public CourseModel selectCourse(String id_course){
19         CourseModel course=
20             restTemplate.getForObject(
21                 "http://localhost:8080/rest/course/view/"+id_course,
22                 CourseModel.class);
23         return course;
24     }
25
26
27     @Override
28     public List<CourseModel> selectAllCourse()
29     {
30         List<CourseModel> course= restTemplate.getForObject("http://localhost:8080/rest/course/viewall", List.class);
31         return course;
32     }
33
34 }
35
36

```

3. Coba akses <http://localhost:8080/rest/course/view/CSC123> untuk lihat course by id



4. Coba jalankan <http://localhost:9090/course/view/CSC123>



Courses by idCourse

ID = CSC123

Nama = PSP

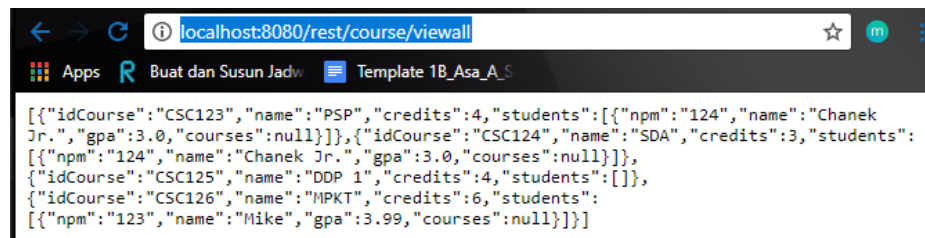
SKS = 4

Mahasiswa yang mengambil

- 124-Chanek Jr.

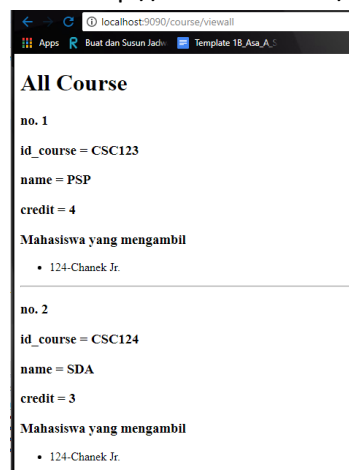
** Bagian viewall-course

akses <http://localhost:8080/rest/course/viewall>. Hasilnya seperti berikut:



```
[{"idCourse": "CSC123", "name": "PSP", "credits": 4, "students": [{"npm": "124", "name": "Chanek Jr.", "gpa": 3.0, "courses": null}], {"idCourse": "CSC124", "name": "SDA", "credits": 3, "students": [{"npm": "124", "name": "Chanek Jr.", "gpa": 3.0, "courses": null}], {"idCourse": "CSC125", "name": "DDP 1", "credits": 4, "students": []}, {"idCourse": "CSC126", "name": "MPKT", "credits": 6, "students": [{"npm": "123", "name": "Mike", "gpa": 3.99, "courses": null}]}
```

akses <http://localhost:9090/course/viewall>. Hasilnya seperti berikut:



```
All Course
no. 1
id_course = CSC123
name = PSP
credit = 4
Mahasiswa yang mengambil
  • 124-Chanek Jr.
no. 2
id_course = CSC124
name = SDA
credit = 3
Mahasiswa yang mengambil
  • 124-Chanek Jr.
```

Write up

Pada tutorial 7 ini saya mempelajari penggunaan producer sebagai backend dan consumer sebagai

pengguna data producer. Tutorial 7 kali ini membutuhkan 2 project yang saling terkait. Untuk dapat membuat project consumer, kita harus membuat producer terlebih dahulu. Di producer, pengerjaan tidak jauh berbeda dengan tutorial sebelumnya, hanya saja web service ini membutuhkan class controller baru yakni restcontroller. @RequestMapping("/rest") pada class header tersebut artinya semua request mapping di dalam kelas tersebut harus selalu diawali dengan "/rest". Method-method di REST Controller mengembalikan objek yang ingin dikembalikan pada service tersebut.

- Spring Boot secara otomatis akan mengembalikan output berupa objek dengan format JSON pada tampilan Web.
- Beberapa object yang dapat dikembalikan misalkan model Class dan Map (key value Map).

Service Consumer yang akan mengonsumsi web service untuk mengakses data-data Student dari Tutorial07Producer, Karena Service Producer dan Service Consumer harus dijalankan secara bersamaan, ubah berkas application.properties yang ada di folder resources dengan menambahkan baris server.port=9090. Nantinya Aplikasi Service Producer akan dijalankan di localhost:8080 sedangkan Service Consumer akan dijalankan di localhost:9090 Kemudian kita butuh interface dan class implementasi dari interface tersebut (yg baru), yakni studentDAO dan courseDAO. Kita menggunakan kelas RestTemplate untuk meng-consume object REST web service. Method yang

digunakan adalah `getForObject` yang menerima parameter berupa URL producer web service dan tipe class dari object yang didapatkan. Selanjutnya, kita ingin mengubah agar `StudentService` mengambil data dari web service bukan dari database. Kita tidak perlu menghapus class `StudentServiceDatabase`. Karena scalable system, kita cukup menambahkan class baru yaitu `StudentServiceRest` yang mengimplement `StudentService` di package service.