

Latihan 1

Pada latihan ini membuat rest membuat code seperti ini pada StudentRestController.

```
@RequestMapping("/student/viewall")
public List<StudentModel> viewall(){
    List<StudentModel> student = studentService.selectAllStudents();
    return student;
}
```

Pada method ini mengembalikan List dari student dimana menggunakan method selectAllStudents untuk mengambil data semua student dari database. Dan pada hasilnya akan menampilkan format JSON seperti berikut ini

The screenshot shows a web browser at the URL `localhost:8080/rest/student/viewall`. The response is a JSON array of student objects, each containing an `npm`, `name`, `gpa`, and a list of `courses`. Below the browser, a JSON viewer shows the parsed JSON structure in two columns: 'String parse' and 'JS eval'.

```
[{"npm": "123", "name": "chanek", "gpa": 3.52, "courses": [{"id_course": "CSC126", "name": "MPKT", "credits": 6, "students": null}]}, {"npm": "124", "name": "aba", "gpa": 3.21, "courses": [{"id_course": "CSC123", "name": "PSP", "credits": 4, "students": null}, {"id_course": "CSC124", "name": "SDA", "credits": 3, "students": null}]}, {"npm": "3445", "name": "Ayu Sari", "gpa": 3.42, "courses": [{"id_course": "CSC123", "name": "PSP", "credits": 4, "students": null}]}]
```

The JSON viewer displays the following structure:

```
{
  "npm": "123",
  "name": "chanek",
  "gpa": 3.52,
  "courses": [
    {
      "id_course": "CSC126",
      "name": "MPKT",
      "credits": 6,
      "students": null
    }
  ]
}, {
  "npm": "124",
  "name": "aba",
  "gpa": 3.21,
  "courses": [
    {
      "id_course": "CSC123",
      "name": "PSP",
      "credits": 4,
      "students": null
    },
    {
      "id_course": "CSC124",
      "name": "SDA",
      "credits": 3,
      "students": null
    }
  ]
}, {
  "npm": "3445",
  "name": "Ayu Sari",
  "gpa": 3.42,
  "courses": [
    {
      "id_course": "CSC123",
      "name": "PSP",
      "credits": 4,
      "students": null
    }
  ]
}]
```

Latihan 2

Membuat CourseRestController yang berisi code seperti berikut ini

```

@RequestMapping("/course/view/{idCourse}")
public CourseModel viewCourse (@PathVariable(value = "idCourse") String idCourse){
    CourseModel course = studentService.selectCourse(idCourse);
    return course;
}

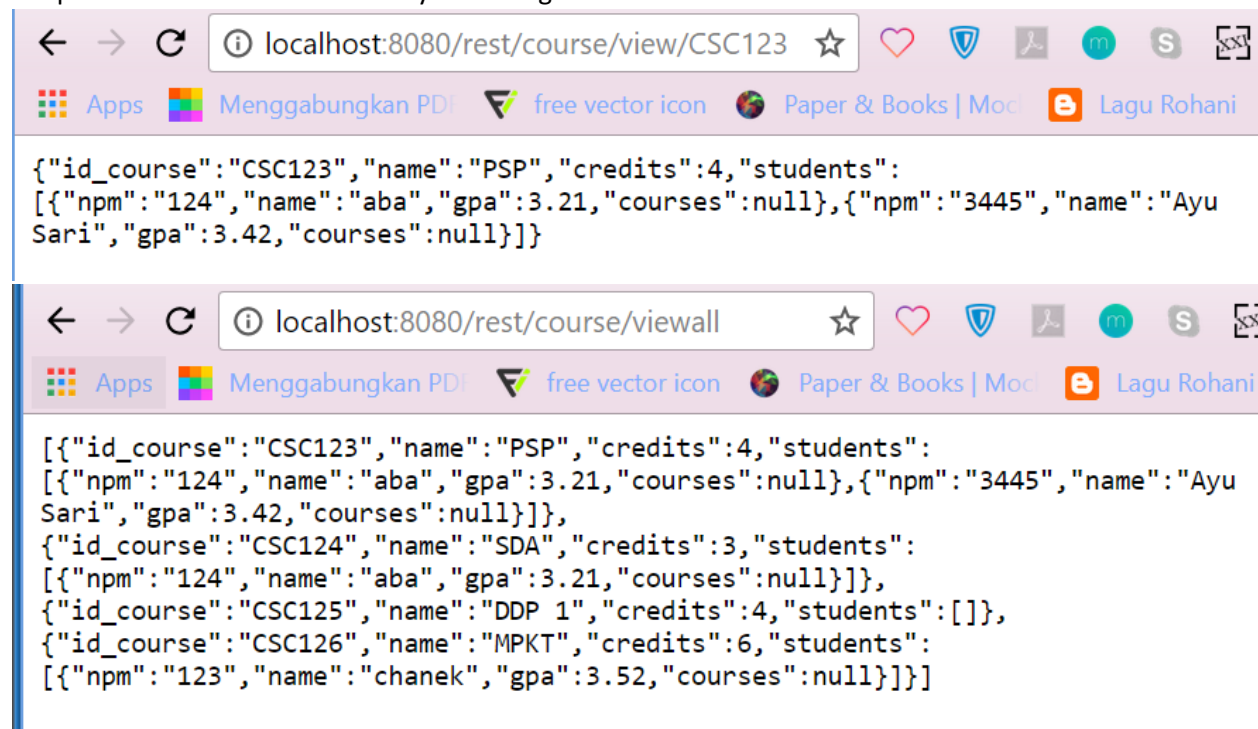
@RequestMapping("/course/viewall")
public List<CourseModel> viewAllCourse (Model model) {
    List<CourseModel> course = studentService.selectAllCourses();
    return course;
}

```

Kode viewCourse tersebut akan menampilkan course sesuai dengan idCourse yang di cari. Pada viewCourse ini menggunakan method selectCourse untuk mengambil course dari database yang ada pada mapper sesuai dengan idCourse yang dicari.

Kode ViewAllCourse menampilkan semua course dan mahasiswa yang mengambil course tersebut. Pada viewAllCourse ini menggunakan method selectAllCourse yang mengambil data pada database untuk menampilkan semua course yang ada dan mahasiswa yang mengambilnya.

Output dari ke dua kode tersebut yaitu sebagai berikut



```

{"id_course":"CSC123","name":"PSP","credits":4,"students":
[{"npm":"124","name":"aba","gpa":3.21,"courses":null},{ "npm":"3445","name":"Ayu
Sari","gpa":3.42,"courses":null}]}

[{"id_course":"CSC123","name":"PSP","credits":4,"students":
[{"npm":"124","name":"aba","gpa":3.21,"courses":null},{ "npm":"3445","name":"Ayu
Sari","gpa":3.42,"courses":null}]},
{"id_course":"CSC124","name":"SDA","credits":3,"students":
[{"npm":"124","name":"aba","gpa":3.21,"courses":null}]},
{"id_course":"CSC125","name":"DDP 1","credits":4,"students":[]},
{"id_course":"CSC126","name":"MPKT","credits":6,"students":
[{"npm":"123","name":"chanek","gpa":3.52,"courses":null}]}]

```

Jika di parsing maka hasilnya seperti berikut ini:

viewCourse

String parse	JS eval
<pre> { "id_course": "CSC123", "name": "PSP", "credits": 4, "students": [{ "npm": "124", "name": "aba", "gpa": 3.21, "courses": null }, { "npm": "3445", "name": "Ayu Sari", "gpa": 3.42, "courses": null }] } </pre>	<pre> { "id_course": "CSC123", "name": "PSP", "credits": 4, "students": [{ "npm": "124", "name": "aba", "gpa": 3.21, "courses": null }, { "npm": "3445", "name": "Ayu Sari", "gpa": 3.42, "courses": null }] } </pre>

viewAll

String parse	JS eval
<pre> [{ "id_course": "CSC123", "name": "PSP", "credits": 4, "students": [{ "npm": "124", "name": "aba", "gpa": 3.21, "courses": null }, { "npm": "3445", "name": "Ayu Sari", "gpa": 3.42, "courses": null }] }, { "id_course": "CSC124", "name": "SDA", "credits": 3, "students": [{ "npm": "124", </pre>	<pre> [{ "id_course": "CSC123", "name": "PSP", "credits": 4, "students": [{ "npm": "124", "name": "aba", "gpa": 3.21, "courses": null }, { "npm": "3445", "name": "Ayu Sari", "gpa": 3.42, "courses": null }] }, { "id_course": "CSC124", "name": "SDA", "credits": 3, "students": [{ "npm": "124", </pre>

Latihan 3

Membuat kode pada service consumer untuk viewAll student pada class StudentServiceRest

```
@Override
public List<StudentModel> selectAllStudents () {
    Log.info ("REST - select all students");
    return studentDAO.selectAllStudents();
}
```

Pada studentDAO.selectAllStudents akan mengarah pada StudentDAOImp dengan isi seperti berikut

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

Method tersebut menggunakan RestTemplate dan method getForObject yang menerima parameter link localhost:8080/rest/student/viewall dan menampilkan list dari student pada producer webservice. Maka output nya adalah seperti berikut

No	NPM	Name	GPA	Cum laude	Delete	Update
1	123	chaneK	3.52	Cum Laude!	Delete Data	Update Data
2	124	aba	3.21	Sangat Memuaskan!	Delete Data	Update Data
3	3445	Ayu Sari	3.42	Sangat Memuaskan!	Delete Data	Update Data

Latihan 4

Membuat consumer untuk CourseModel. Pada CourseServiceRest berisi seperti berikut

```
@Slf4j
@Service
@Primary
public class CourseServiceRest implements CourseService{

    @Autowired
    private CourseDAO courseDAO;

    @Override
    public CourseModel selectCourse(String idCourse) {
        Log.info("REST - select course student with id {}", idCourse);
        return courseDAO.selectCourse(idCourse);
    }

    @Override
    public List<CourseModel> selectAllCourse(){
        Log.info("REST - select all course");
        return courseDAO.selectAllCourse();
    }
}
```

Terdapat method `selectCourse` untuk melihat course sesuai dengan `idCourse` dan menggunakan method `selectCourse` yang ada pada `courseDAO`. Begitu juga pada method `selectAllCourse` untuk menampilkan semua course yang ada pada `courseDAO`.

```
package com.example.dao;

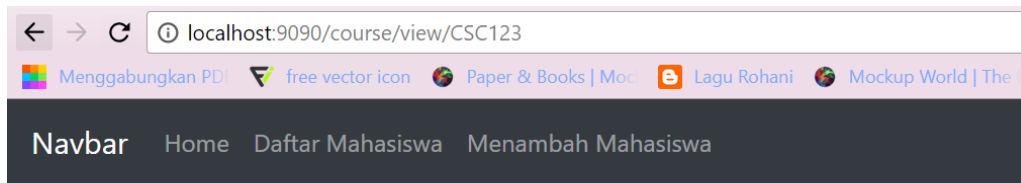
import java.util.List;

public interface CourseDAO {
    CourseModel selectCourse(String idCourse);
    List<CourseModel> selectAllCourse();
}
```

Pada `studentDAO` akan mengarah pada implementasinya pada `CourseDAOImpl` yang berisi seperti berikut ini

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

Pada class ini menggunakan `restTemplate` dan method `getForObject` yang menerima parameter url untuk mengembalikan object course dari producer webservice. Pada consumer tidak perlu mengakses database lagi karena sudah dilakukan oleh producer. Tampilan dari kedua method tersebut adalah sebagai berikut:



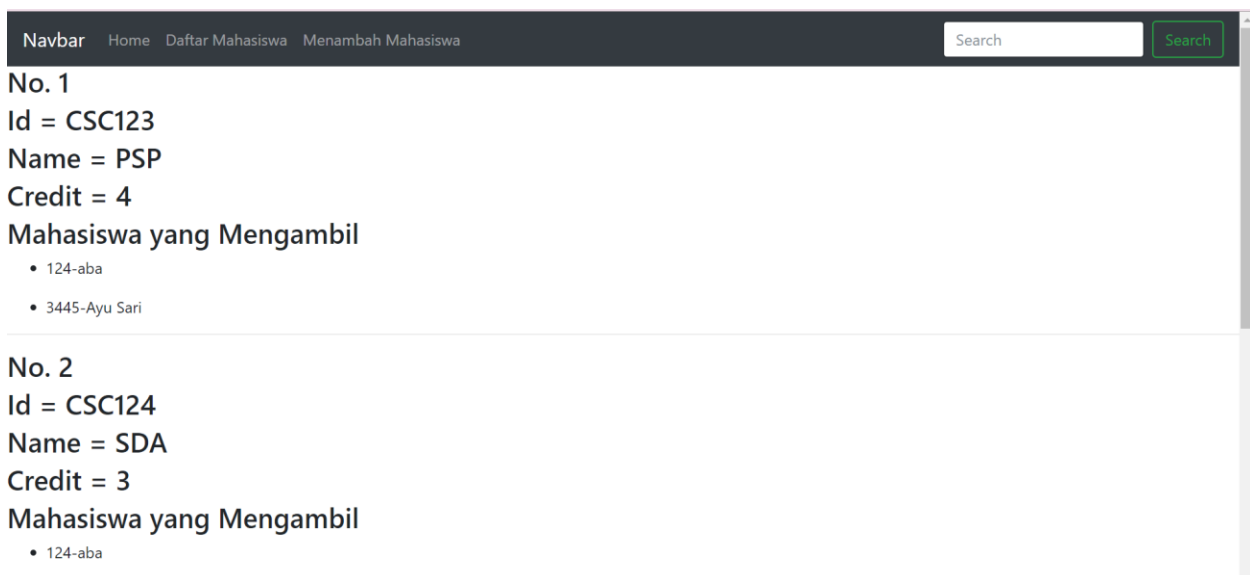
ID = CSC123

Nama = PSP

SKS = 4

Mahasiswa yang mengambil

- 124-aba
- 3445-Ayu Sari



Pada console terlihat log.info yang dijalankan oleh consumer yaitu sebagai berikut

```
Tutorial07Consumer - Tutorial07Consumer [Spring Boot App] C:\Program Files\Java\jre1.8.0_144\bin\javaw.exe (Nov 4, 2017, 12:43:52 PM)
-- [nio-9090-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring FrameworkServlet 'dispatcherServlet'
-- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet      : FrameworkServlet 'dispatcherServlet': initialization started
-- [nio-9090-exec-1] o.s.web.servlet.DispatcherServlet      : FrameworkServlet 'dispatcherServlet': initialization complete
-- [nio-9090-exec-1] com.example.service.StudentServiceRest  : REST - select all students
-- [nio-9090-exec-4] com.example.service.CourseServiceRest   : REST - select course student with id CSC123
-- [nio-9090-exec-7] com.example.service.CourseServiceRest   : REST - select all course
```

Pada tutorial ini yang saya pelajari adalah mengenai webservice yang dapat membuat aplikasi dari beberapa server port bisa terkoneksi yang dimana pada tutorial ini producer sebagai layer back-end dan consumer sebagai layer front-end. Producer akan mengatur akses database sesuai dengan method dan apa yang harus dilakukan sementara pada consumer akan menampilkan apa yang sudah dilakukan oleh producer.

Pada tutorial ini, data pada webservice di producer yaitu berupa JSON dan dapat dikembalikan dengan membuat rest controller class dan akan mengambil object dari database dan ditampilkan ke web. Pada service consumer dapat mengambil data dengan menggunakan RestTemplate class yang akan digunakan untuk mengambil object REST web service dengan method getForObject yang menerima parameter berupa url dan mengarah pada URL producer. Sehingga nantinya consumer dapat menampilkan data tanpa mengakses database.