

## WRITE UP

Pada tutorial ketujuh kali ini, saya mempelajari hal baru terkait Web Service pada Spring Boot. Saya mendapat pengetahuan untuk melakukan pemisahan *layer backend (service producer)* dan *frontend (service consumer)*. *Service consumer* merupakan aplikasi yang langsung berinteraksi dengan *user*. Sedangkan *service producer* merupakan aplikasi yang memberikan data kepada *service consumer* berdasarkan apa yang diminta oleh *service consumer*. Pemisahan tersebut bertujuan agar aplikasi *service consumer* dapat fokus untuk menyediakan dan mengolah data saja ke pengguna tanpa perlu memiliki database. Sedangkan *service producer* hanya bertanggung jawab untuk menyediakan data dari database dan biasanya tidak memiliki *view* yang dapat dilihat user. Pada tutorial ini saya diminta untuk membuat *web service producer* untuk mengakses data-data *Student* yang telah dibuat pada tutorial sebelumnya. Untuk membuat *producer*, kita harus membuat sebuah controller baru dengan menggunakan REST web service, caranya dengan menambahkan anotasi `@RestController` dan `@RequestMapping("/rest")` pada *class header*. Kemudian, *view* dari *producer* akan mengembalikan *object* `StudentModel` dalam format JSON.

Setelah membuat *web service producer*, kemudian saya membuat *web Service Consumer* yang akan mengonsumsi web service untuk mengakses data-data *Student* dari *producer*. Karena *Service Producer* dan *Service Consumer* harus dijalankan secara bersamaan, saya mendapat pengetahuan baru untuk mengubah berkas *application.properties* *Consumer* yang ada di folder *resources* dengan menambahkan `server.port` yang berbeda dengan *Producer* (9090). Untuk membuat *Consumer* kita harus menambahkan *interface* baru dengan nama `StudentDAO` dan mengimplementasi kelas `StudentDAO` tersebut dengan nama kelas `StudentDAOImpl.java`. Pada kelas `StudentDAOImpl.java` saya mendapatkan pengetahuan baru untuk menggunakan method `getForObject` yang menerima parameter berupa URL *producer web service* dan tipe *class* dari *object* yang didapatkan. Setelah itu, saya menambahkan class baru yaitu `StudentServiceRest` yang mengimplement `StudentService`.

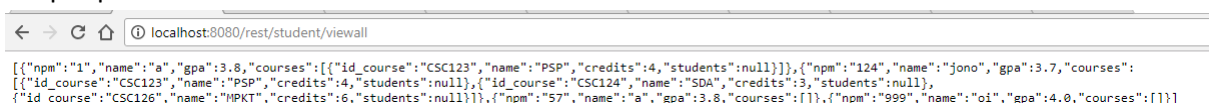
### Latihan

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"`.

- Menambahkan request mapping untuk `/student/viewall` pada *student rest controller*

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

- Output pada localhost



```
localhost:8080/rest/student/viewall
[{"npm": "1", "name": "a", "gpa": 3.8, "courses": [{"id_course": "CSC123", "name": "PSP", "credits": 4, "students": null}], {"npm": "124", "name": "Jono", "gpa": 3.7, "courses": [{"id_course": "CSC123", "name": "PSP", "credits": 4, "students": null}, {"id_course": "CSC124", "name": "SDA", "credits": 3, "students": null}, {"id_course": "CSC126", "name": "MPKT", "credits": 6, "students": null}], {"npm": "57", "name": "a", "gpa": 3.8, "courses": []}, {"npm": "999", "name": "oi", "gpa": 4.0, "courses": []}]
```

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).

- Membuat class course rest controller yang berikan method untuk melihat course dan melihat semua course / view all.

```
1 package com.example.rest;
2
3 import java.util.List;
4
5 @RestController
6 @RequestMapping("/rest")
7 public class CourseRestController {
8     @Autowired
9     StudentService studentService;
10
11     @RequestMapping("/course/view/{id_course}")
12     public CourseModel viewCourse (@PathVariable(value = "id_course") String id_course)
13     {
14         CourseModel course = studentService.selectCoursesStudent(id_course);
15         return course;
16     }
17
18     @RequestMapping("/course/viewall")
19     public List<CourseModel> viewAllCourse ()
20     {
21         List<CourseModel> courses = studentService.selectAllCourses();
22         return courses;
23     }
24 }
```

- Membuat request mapping untuk course/viewall pada student controller

```
@RequestMapping("/course/viewall")
public String viewAllCourse (Model model)
{
    List<CourseModel> courses = courseDAO.selectAllCourses();
    model.addAttribute ("courses", courses);

    return "viewAllCourse";
}
```

- Membuat html untuk view all course

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<link rel = "stylesheet" href = "/css/bootstrap.min.css"/>
<title>View All Courses</title>
</head>
<body>
<div th:replace="fragments/fragment :: header"></div>
<h1>All Courses</h1>

<div th:each="course,iterationStatus: ${courses}">
<h3 th:text="'No. ' + ${iterationStatus.count}">No. 1</h3>
<h3 th:text="'Id course = ' + ${course.id_course}">Student NPM</h3>
<h3 th:text="'Name = ' + ${course.name}">Student Name</h3>
<h3 th:text="'Credit = ' + ${course.credits}">Student GPA</h3>

<h3>Mahasiswa yang mengambil</h3>
<ul th:each="student,iterationStatus: ${course.students}">
<li th:text="${student.npm} + '-' + ${student.name}">
    Nama kuliah-X sks
</li>
</ul>
</div>
<div th:replace="fragments/fragment :: footer"></div>
</body>
</html>
```

- Membuat method select all courses pada student mapper

```
    @Select("select id_course, name, credits from course")  
    @Results(value = {  
        @Result(property="id_course", column="id_course"),  
        @Result(property="name", column="name"),  
        @Result(property="credits", column="credits"),  
        @Result(property="students", column="id_course",  
            javaType = List.class,  
            many=@Many(select="selectStudentCourses"))  
    })  
    List<CourseModel> selectAllCourses();
```

- Output course/viewall pada localhost



- Output rest/course/viewall pada localhost

```
localhost:8080/rest/course/viewall  
[{"id_course":"CSC123","name":"PSP","credits":4,"students":[{"npm":"1","name":"a","gpa":3.8,"courses":null},{"npm":"124","name":"jono","gpa":3.7,"courses":null}],  
{"id_course":"CSC124","name":"SDA","credits":3,"students":[{"npm":"124","name":"jono","gpa":3.7,"courses":null}],{"id_course":"CSC125","name":"DOP 1","credits":4,"students":[]},  
{"id_course":"CSC126","name":"MPKT","credits":6,"students":[{"npm":"124","name":"jono","gpa":3.7,"courses":null}]}
```

- Output rest/course/view/csc123 pada localhost

```
localhost:8080/rest/course/view/csc123  
{"id_course":"CSC123","name":"PSP","credits":4,"students":[{"npm":"1","name":"a","gpa":3.8,"courses":null},{"npm":"124","name":"jono","gpa":3.7,"courses":null}]}
```

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

- Melengkapi method selectAllStudents yang ada di kelas StudentServiceRest

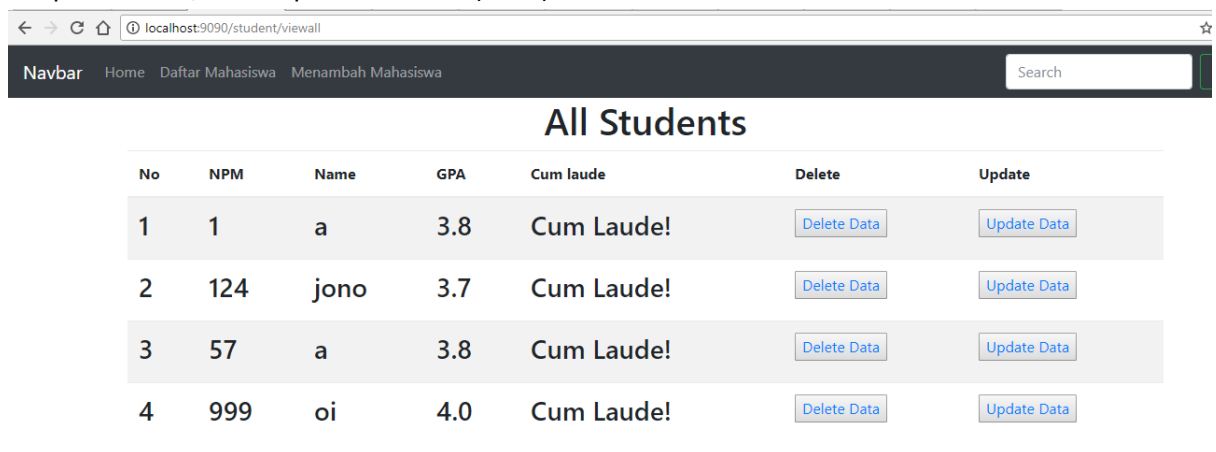
```
@Override
public List<StudentModel> selectAllStudents ()
{
    log.info("REST - select all students");
    return studentDAO.selectAllStudents();
}
```

- Melengkapi method selectAllStudents pada class StudentDAOImpl.java

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

    return students;
}
```

- Output student/viewall pada localhost (9090)



No	NPM	Name	GPA	Cum laude	Delete	Update
1	1	a	3.8	Cum Laude!	<a href="#">Delete Data</a>	<a href="#">Update Data</a>
2	124	jono	3.7	Cum Laude!	<a href="#">Delete Data</a>	<a href="#">Update Data</a>
3	57	a	3.8	Cum Laude!	<a href="#">Delete Data</a>	<a href="#">Update Data</a>
4	999	oi	4.0	Cum Laude!	<a href="#">Delete Data</a>	<a href="#">Update Data</a>

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

- Membuat class CourseDao service yang berisikan method selectAllcourses dan selectCoursesStudent

```
package com.example.dao;

import java.util.List;
import org.apache.ibatis.annotations.Param;
import com.example.model.CourseModel;

public interface CourseDAO {
    CourseModel selectCoursesStudent (String id_course);
    List<CourseModel> selectAllCourses();
}
```

- Membuat *class* CourseDAOImpl yang mengimplementasikan CourseDAO. *Class* tersebut berisikan method selectAllCourses dan selectCoursesStudent.

```
1
2
3
4
5 @Service
6 public class CourseDAOImpl implements CourseDAO{
7     @Autowired
8     private RestTemplate restTemplate;
9
10
11     @Bean
12     public RestTemplate restTemplate() {
13         return new RestTemplate();
14     }
15
16     @Override
17     public CourseModel selectCoursesStudent (String id_course)
18     {
19         CourseModel course = restTemplate.getForObject(
20             "http://localhost:8080/rest/course/view/"+id_course,
21             CourseModel.class);
22         return course;
23     }
24
25     @Override
26     public List<CourseModel> selectAllCourses()
27     {
28         List<CourseModel> courses = restTemplate.getForObject(
29             "http://localhost:8080/rest/course/viewall",
30             List.class);
31
32         return courses;
33     }
34 }
35 }
```

- Membuat interface course service yang berisikan method selectAllcourses dan selectCoursesStudent

```
1 package com.example.service;
2
3 import java.util.List;
4
5 import com.example.model.CourseModel;
6
7 public interface CourseService {
8     List<CourseModel> selectAllCourses();
9
10     CourseModel selectCoursesStudent(String id_course);
11 }
12 }
```

- Membuat *class* `courseServiceRest` yang mengimplementasikan `courseService`. *Class* tersebut berisikan method `selectAllCourses` dan `selectCoursesStudent`.

```
public class CourseServiceRest implements CourseService {  
  
    @Autowired  
    private CourseDAO courseDAO;  
  
    @Override  
    public List<CourseModel> selectAllCourses ()  
    {  
        log.info("REST - select all courses");  
        return courseDAO.selectAllCourses();  
    }  
  
    @Override  
    public CourseModel selectCoursesStudent(String id_course)  
    {  
        log.info ("select course");  
        return courseDAO.selectCoursesStudent(id_course);  
    }  
}
```

- Memodifikasi method `course/view/id_course` pada `studentController` dengan mengganti `studentDAO` menjadi `courseDAO`

```
@RequestMapping("/course/view/{id_course}")  
public String viewCourse (Model model, @PathVariable(value = "id_course") String id_course)  
{  
    CourseModel course = courseDAO.selectCoursesStudent(id_course);  
    if (course != null) {  
        model.addAttribute ("course", course);  
        return "viewCourse";  
    } else {  
        model.addAttribute ("id_course", id_course);  
        return "not-found-2";  
    }  
}
```

- Menambahkan method `viewAllCourse` pada `student controller`

```
@RequestMapping("/course/viewall")  
public String viewAllCourse (Model model)  
{  
    List<CourseModel> courses = courseDAO.selectAllCourses();  
    model.addAttribute ("courses", courses);  
  
    return "viewAllCourse";  
}
```

- Menambahkan view all course html

```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3   <head>
4     <link rel = "stylesheet" href = "/css/bootstrap.min.css"/>
5     <title>View All Courses</title>
6   </head>
7   <body>
8     <div th:replace="fragments/fragment :: header"></div>
9     <h1>All Courses</h1>
10
11    <div th:each="course,iterationStatus: ${courses}">
12      <h3 th:text="'No. ' + ${iterationStatus.count}">No. 1</h3>
13      <h3 th:text="'Id course = ' + ${course.id_course}">Student NPM</h3>
14      <h3 th:text="'Name = ' + ${course.name}">Student Name</h3>
15      <h3 th:text="'Credit = ' + ${course.credits}">Student GPA</h3>
16
17      <h3>Mahasiswa yang mengambil</h3>
18      <ul th:each="student,iterationStatus: ${course.students}">
19        <li th:text="${student.npm} + '-' + ${student.name} " >
20          Nama kuliah-X sks
21        </li>
22      </ul>
23    </div>
24    <div th:replace="fragments/fragment :: footer"></div>
25  </body>
26 </html>
27
```

- Output course/view/csc123 pada localhost (9090)

**Navbar** Home Daftar Mahasiswa Menambah Mahasiswa

**ID = CSC123**

**Name = PSP**

**SKS = 4**

**Mahasiswa yang mengambil**

- 1-a
- 124-jono

---

**Mata Kuliah APAP**

- Output course/viewall pada localhost (9090)

Navbar Home Daftar Mahasiswa Menambah Mahasiswa

## All Courses

No. 1

Id course = CSC123

Name = PSP

Credit = 4

Mahasiswa yang mengambil

- 1-a
- 124-jono

No. 2

Id course = CSC124

Name = SDA

Credit = 3

Mahasiswa yang mengambil

- 124-jono