

Tutorial 7

Membuat Web Service Menggunakan Spring Boot Framework

Write-up:

Pada tutorial ini dijelaskan mengenai *web service* yang dapat membuat aplikasi-aplikasi yang tersebar pada beberapa *server* dapat saling berkomunikasi. Contoh pada tutorial kali ini adalah dengan *service producer* yang menangani *layer backend* dan *service consumer* yang menangani *layer frontend*. *Service producer* akan mengakses *database* untuk mengambil data yang diinginkan yang kemudian akan ditampilkan ke halaman *web* melalui *service consumer*. Untuk dapat melakukan hal tersebut, *service producer* menyediakan *web service* yang dapat dikonsumsi oleh *service consumer*. *Web service* merupakan URL yang akan mengembalikan data dalam representasi seperti JSON atau XML.

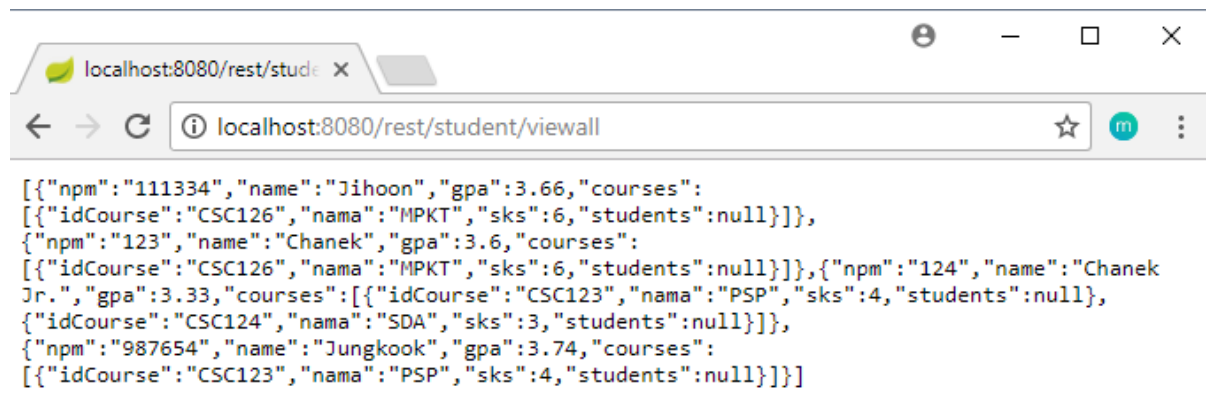
Pada tutorial ini, data yang dikembalikan oleh *web service* adalah JSON. Data tersebut dapat dikembalikan dengan membuat *rest controller class* pada *package* yang terpisah dari *package controller* biasa, yang akan mengambil objek dari *database* dan menampilkan ke tampilan *web* berupa JSON dari data tersebut. Kemudian, *service consumer* dapat mengambil data tersebut menggunakan *RestTemplate class* yang mana digunakan untuk meng-*consume* objek REST *web service* dengan method *getForObject* yang menerima parameter berupa URL *producer web service* dan tipe *class* dari objek yang didapatkan. Sehingga, *service consumer* dapat menampilkan data tanpa perlu tahu bagaimana cara mengakses *database*.

Latihan 1: Service yang mengembalikan seluruh *student* pada *database*

Untuk membuat *service* yang mengembalikan data seluruh *student* pada *database*, dibuat *RequestMapping /rest/student/viewall* pada *controller* *StudentRestController.java* sebagai berikut:

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

RequestMapping yang dituliskan hanya */student/viewall* karena pada *class header* sudah dituliskan *@RequestMapping("/rest")* yang mana berarti untuk semua *RequestMapping* pada setiap *method* yang ada di *class* tersebut selalu diawali dengan */rest*. Kemudian, *method* tersebut akan mengambil *list of students* pada *database* dengan memanggil *method* *selectAllStudents* pada *StudentService* yang kemudian akan memanggil *method* *selectAllStudents* pada *StudentMapper class* yang mana akan mengambil data semua *student* pada *database*. Ketika objek *list of students* tersebut sudah didapatkan, maka dengan *controller rest*, Spring Boot akan menampilkan *output* berupa objek dengan format JSON pada tampilan *web*, yaitu sebagai berikut:



Latihan 2: Service untuk class Course

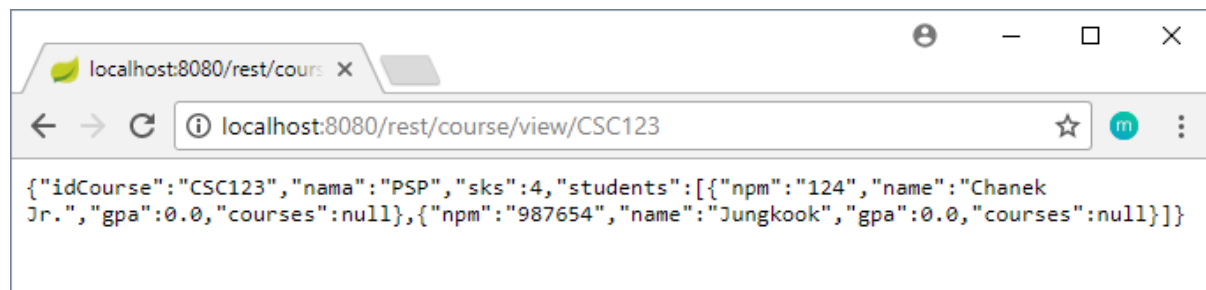
Pertama, dibuat *rest controller* untuk *course* sebagai berikut:

```
@RestController
@RequestMapping("/rest")
public class CourseRestController {

    @Autowired
    CourseService courseService;

    @RequestMapping("/course/view/{id}")
    public CourseModel viewCourse (Model model,
        @PathVariable(value = "id") String id)
    {
        CourseModel course = courseService.selectCourse (id);
        return course;
    }
}
```

Untuk *service* yang mengembalikan suatu *course* berdasarkan *id*, dibuat *method* pada *controller* tersebut dengan *RequestMapping* `/course/view/{id}` yang mana akan mengambil parameter *id* *course* yang ingin dilihat pada *path variable* url. Kemudian, *method* akan mengambil data *course* yang diinginkan dengan memanggil *method* `selectCourse` pada *CourseService* class. *Method* `selectCourse` tersebut kemudian akan mengambil data *course* yang diinginkan dengan memanggil *method* `selectCourse` pada *StudentMapper* class yang mana akan mengambil data *course* yang ada di *database*. Sehingga, *controller rest method* `viewCourse` dapat mengambil objek *course* yang diinginkan dan akan ditampilkan pada tampilan web dalam bentuk JSON sebagai berikut:



Untuk *service* yang mengembalikan semua *course* pada *database*, dibuat *method* pada *CourseController* sebagai berikut:

```
@RequestMapping("/course/viewall")
public List<CourseModel> viewAll() {
    List<CourseModel> courses = courseService.selectAllCourses();
    return courses;
}
```

Method tersebut akan menerima *RequestMapping* */course/viewall* yang mana akan memanggil *method* *selectAllCourses* pada *CourseService* *class* yang akan memanggil *selectAllCourses* pada *CourseMapper* *class*, sehingga *list of courses* dapat diambil dari *database* dan ditampilkan pada tampilan web dalam bentuk JSON oleh *rest controller* sebagai berikut:



Latihan 3: *Service consumer* untuk melihat semua *student*.

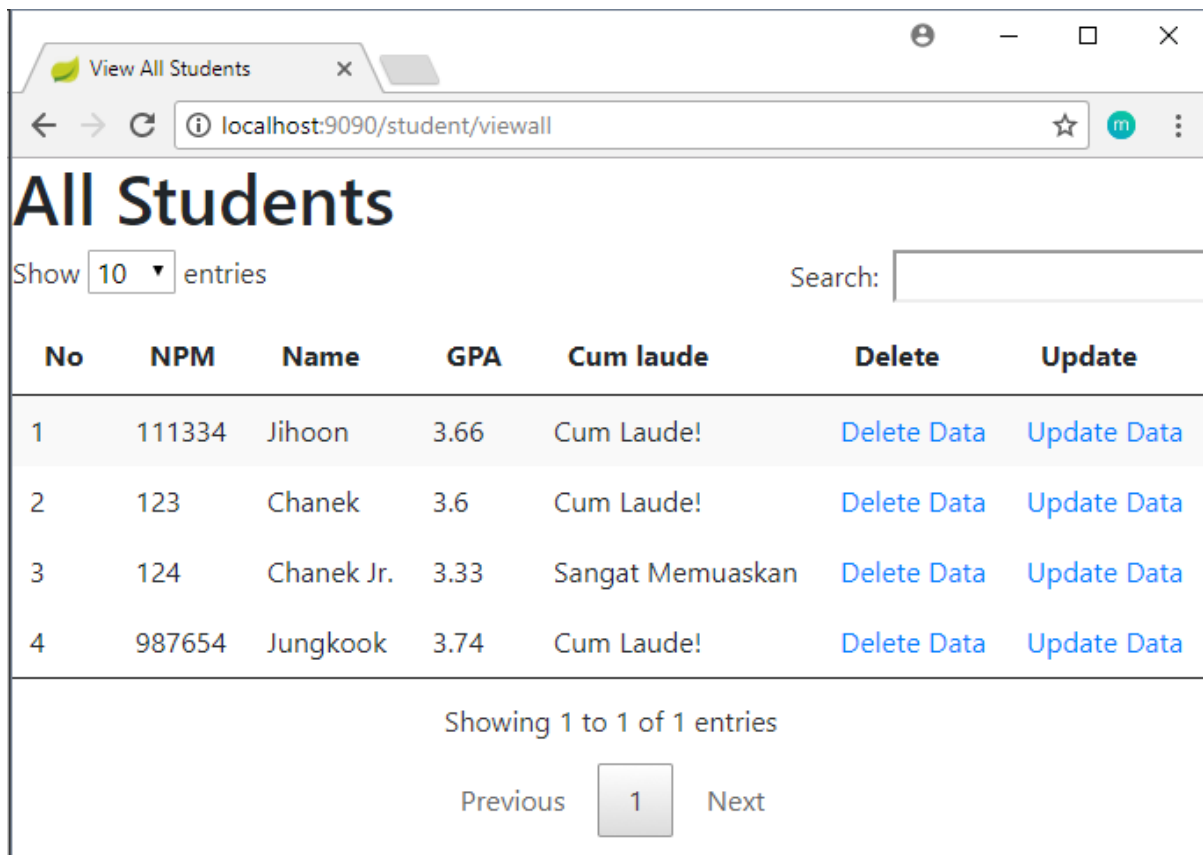
Method yang terdapat pada *StudentServiceRest* *class* adalah sebagai berikut:

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

Method tersebut akan dijalankan ketika *user* memanggil url */student/viewall* karena pada *class* *StudentServiceRest* tersebut sudah diberikan anotasi pada *class header* yaitu *@Primary*, yang mana akan menginstansiasi *StudentService* *class* menggunakan *StudentServiceRest* *class*. Kemudian, *method* tersebut akan memanggil *method* *selectAllStudents* pada *studentDAO* *class* yang diimplementasikan pada *StudentDAOImpl* *class* sebagai berikut:

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

Method tersebut menggunakan RestTemplate class untuk meng-consume object REST web service. Method yang digunakan pada RestTemplate class tersebut adalah getForObject yang mana menerima parameter berupa URL producer web service /rest/student/viewall yang akan mengembalikan list of students dari producer web service. Sehingga, method selectAllStudents dapat menerima data list of students tanpa perlu mengakses database. Data list of students akan diterima oleh controller dan dapat ditampilkan pada tampilan web sebagai berikut:



No	NPM	Name	GPA	Cum laude	Delete	Update
1	111334	Jihoon	3.66	Cum Laude!	Delete Data	Update Data
2	123	Chanek	3.6	Cum Laude!	Delete Data	Update Data
3	124	Chanek Jr.	3.33	Sangat Memuaskan	Delete Data	Update Data
4	987654	Jungkook	3.74	Cum Laude!	Delete Data	Update Data

Showing 1 to 1 of 1 entries

Previous 1 Next

Latihan 4: Service consumer untuk class CourseModel.

Service consumer untuk melihat course berdasarkan id, controller /course/view/{id} akan memanggil method selectCourse yang ada pada CourseService class yang diimplementasikan pada CourseServiceRest class sebagai berikut:

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

    @Autowired
    private CourseDAO courseDAO;

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

Method `selectCourse` tersebut akan memanggil `selectCourse` pada `CourseDAO` class yang diimplementasikan pada `CourseDAOImpl` class sebagai berikut:

```
1 package com.example.dao;
2
3 import java.util.List;
4
5 public interface CourseDAO {
6
7     CourseModel selectCourse(String id);
8     List<CourseModel> selectAllCourses();
9 }
10
11 public class CourseDAOImpl implements CourseDAO {
12
13     @Autowired
14     private RestTemplate restTemplate;
15
16     @Override
17     public CourseModel selectCourse(String id)
18     {
19         CourseModel course =
20             restTemplate.getForObject(
21                 "http://localhost:8080/rest/course/view/"+id,
22                 CourseModel.class);
23         return course;
24     }
25 }
```

Method tersebut akan menggunakan `getForObject` method dari `RestTemplate` class yang akan menerima parameter url `/rest/course/view/id` yang akan mengembalikan objek `course` berdasarkan id yang diinginkan dari *producer web service* sehingga *consumer* tidak perlu mengakses *database*. Kemudian, *controller* akan menerima objek tersebut dan dapat ditampilkan pada tampilan web sebagai berikut:



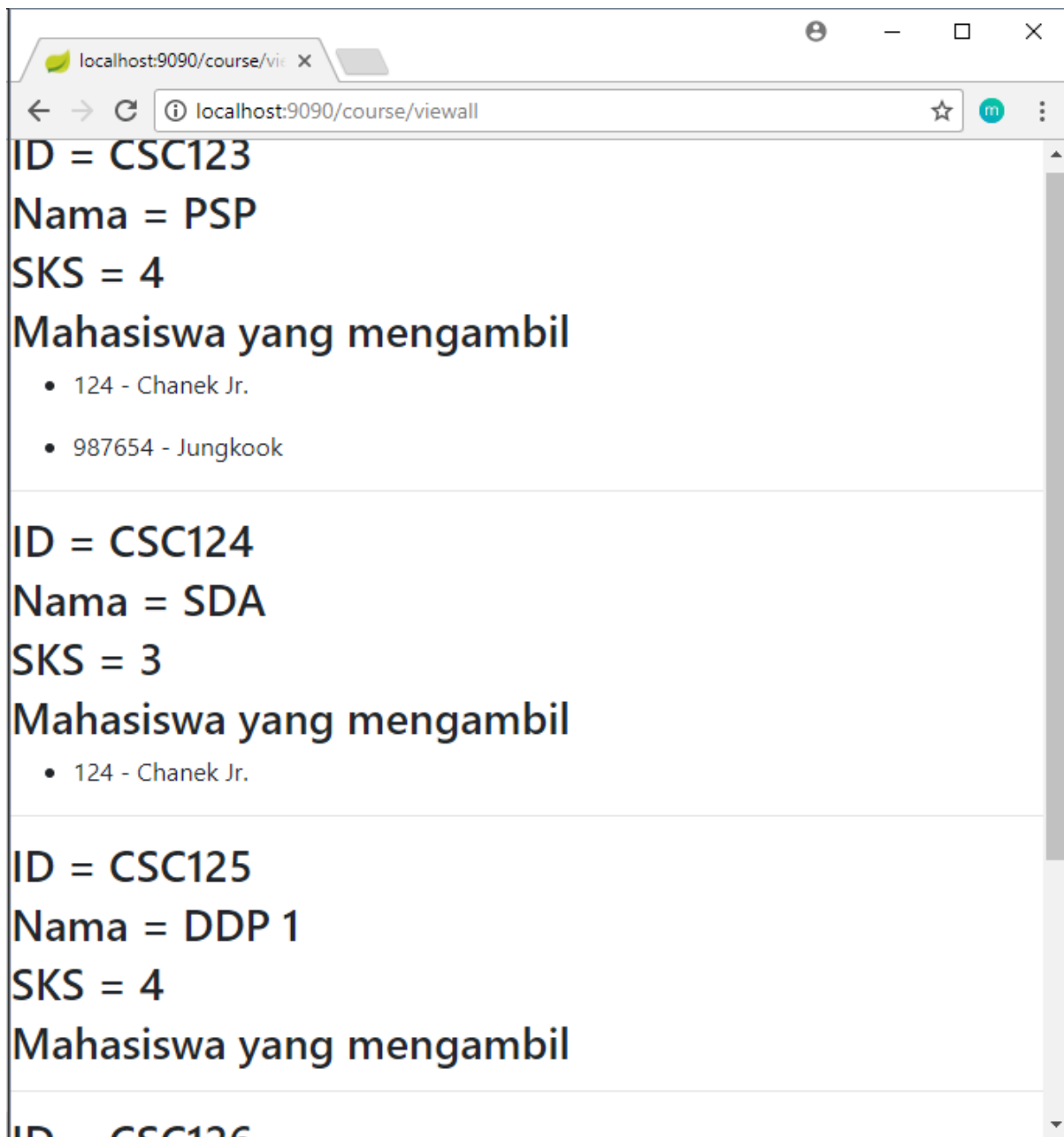
Untuk *service consumer* melihat semua *courses*, *controller* akan memanggil *method* `selectAllCourses` pada *CourseService* *class* yang diimplementasikan pada *CourseServiceRest* sebagai berikut:

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

Method tersebut akan memanggil *method* `selectAllCourses` pada *CourseDAO* *class* yang diimplementasikan pada *CourseDAOImpl* *class* sebagai berikut:

```
@Override
public List<CourseModel> selectAllCourses ()
{
    List<CourseModel> courses =
        restTemplate.getForObject(
            "http://localhost:8080/rest/course/viewall",
            List.class);
    return courses;
}
```

Method tersebut akan mengambil objek *list of courses* dari *producer web service* dengan url `/rest/course/viewall`, sehingga *consumer* tidak perlu mengambil data dari *database*. Kemudian, *controller* akan memanggil *viewall template* untuk menampilkan ke halaman *web* sebagai berikut:



Karena pada setiap eksekusi terdapat log.info untuk memastikan *method* pada kelas ServiceRest dijalankan, bukan *method* pada kelas ServiceDatabase yang sama-sama mengimplementasikan kelas Service, maka pada *console* akan terlihat sebagai berikut:

```
-- [nio-9090-exec-1] com.example.service.StudentServiceRest : REST - select student with npm 123
-- [nio-9090-exec-6] com.example.service.StudentServiceRest : REST - select all students
-- [nio-9090-exec-3] com.example.service.CourseServiceRest : REST - select course with id CSC123
-- [nio-9090-exec-7] com.example.service.CourseServiceRest : REST - select all courses
```