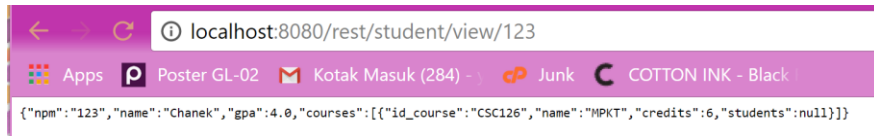


## Write Up Tutorial 7

### Membuat Service producer

Tampilan JSON ketika mengakses path <http://localhost:8080/rest/student/view/123>



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"`.

Membuat *method* `viewAll` pada *class* `StudentRestController` dan akan mengembalikan `List<StudentModel>` dari pemanggilan *method* `selectAllStudents` dari `StudentService`. *Method* `selectAllStudents` dari `StudentService` akan memanggil *method* `selectAllStudent` pada `StudentMapper` yang akan mengambil `students` dan dikumpulkan dalam bentuk *list* dari *database*. *Method* ini dapat diakses dengan path `"/rest/student/viewall"` karena pada *class* `StudentRestController` terdapat `@RequestMapping` `"/rest"` pada header *class* dan mencantumkan `@RequestMapping` `"/student/viewall"` sebelum *method* `viewAll`.

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

Hasil tampilan dengan mengakses path <http://localhost:8080/rest/student/viewall>





- Latihan 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).

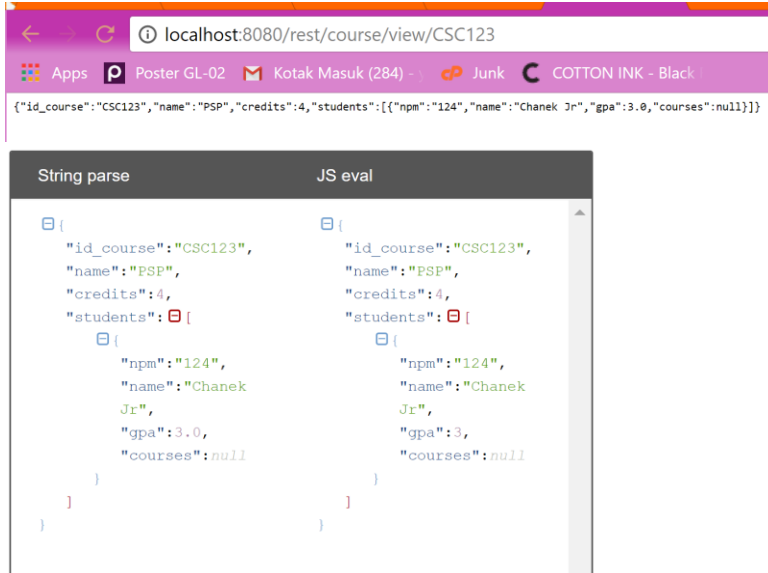
Membuat rest controller untuk course.

```
Tutorial07Producer src main java com example rest CourseRestController
StudentRestController.java CourseRestController.java application.properties CourseMapper.java
1 package com.example.rest;
2
3 import com.example.model.CourseModel;
4 import com.example.service.CourseService;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.web.bind.annotation.PathVariable;
7 import org.springframework.web.bind.annotation.RequestMapping;
8 import org.springframework.web.bind.annotation.RestController;
9
10 import java.util.List;
11
12 @RestController
13 @RequestMapping("/rest")
14 public class CourseRestController {
15     @Autowired
16     CourseService courseService;
17
18     @RequestMapping("/course/view/{id_course}")
19     public CourseModel view(@PathVariable(value = "id_course") String id_course) {
20         CourseModel course = courseService.selectCourse(id_course);
21         return course;
22     }
23 }
```

Mencantumkan `@RequestMapping ("/rest")` agar dapat diakses sesuai dengan kasus Dan pada masing-masing case dicantumkan pula `@RequestMapping` yang sesuai

Untuk dapat melihat course dengan id, dibuatlah *method* view yang akan mengembalikan course berdasarkan id yang terdapat dalam path melalui penggunaan `@PathVariable`. *Method* view akan memanggil *method* `selectCourse` yang terdapat dalam `CourseService` kemudian *method* `selectCourse` yang terdapat pada `CourseService` akan memanggil *method* `selectCourse` yang terdapat dalam `CourseMapper`. Sehingga *method* view pada `CourseRestController` dapat mengambil *object* course berdasarkan id.

Tampilan saat mengakses <http://localhost:8080/rest/course/view/CSC123> (view by ID)



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

Untuk dapat melihat semua course yang terdapat pada database, dibuatlah *method* *viewAll* yang akan mengembalikan *List* dari *object* *course*. *Method* *viewAll* akan memanggil *method* *selectAllCourses* yang terdapat dalam *CourseService* kemudian *method* *selectAllCourses* yang terdapat pada *CourseService* akan memanggil *method* *selectAllCourses* yang terdapat dalam *CourseMapper*. Sehingga *method* *viewAll* pada *CourseRestController* dapat mengambil *List* dari *object* *course*.

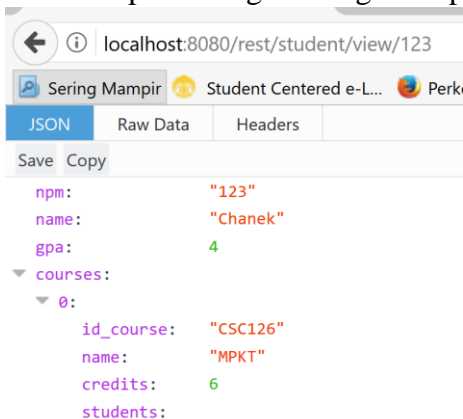
Tampilan saat mengakses <http://localhost:8080/rest/course/viewall> (view by all)



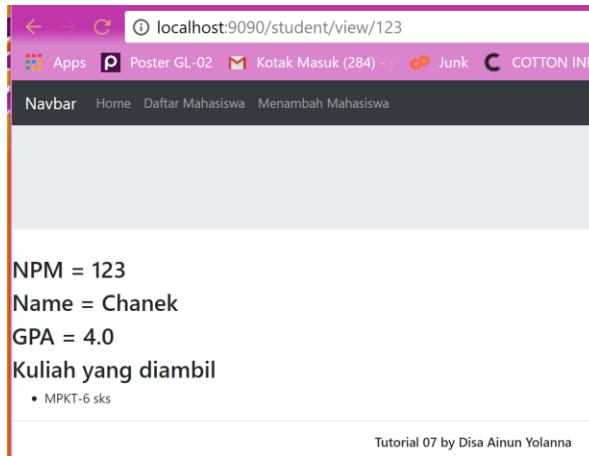


## Membuat Service Consumer

Hasil tampilan dengan mengakses path localhost:8080/student/view/123



Hasil tampilan dengan mengakses path localhost:9090/student/view/123



1. **Latihan 3:** Implementasikan service consumer untuk view all Students dengan melengkapi *method* `selectAllStudents` yang ada di kelas `StudentServiceRest`. Cantumkan dan jelaskan *method* Anda pada write-up Commit dan Push pekerjaan Anda.

Melengkapi `selectAllStudents` yang terdapat pada *class* `StudentServiceRest`

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

*Method* `selectAllStudents` akan dieksekusi ketika user mengakses path `localhost:9090/student/viewall` yang dihandle di `StudentController`. Hal ini terjadi dikarenakan dalam `StudentServiceRest` terdapat `@Primary` yang mana `Autowired` akan menginstansiasi *class* `StudentService` menggunakan `StudentServiceRest`. *Method* `selectAllStudents` akan memanggil *method* `selectAllStudents` pada interface `StudentDAO` yang diimplementasi oleh *class* `StudentDAOImpl`.

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

*Method* diatas merupakan *method* `selectAllStudents` yang terdapat pada *class* `StudentDAOImpl`. *Method* tersebut akan mengembalikan *List of* students tanpa mengakses database. Hal ini terjadi, karena penggunaan kelas `RestTemplate` adalah untuk mengkonsumsi objek REST web service dengan menggunakan `getForObject` yang akan menerima parameter berupa url producer web service yaitu "`http://localhost:8080/rest/student/viewall`" dan mendapat *list of object* students. *List*

tersebut akan diterima oleh StudentController dan akan ditampilkan seperti berikut,

No	NPM	Name	GPA	Cum laude	Delete	Update
No. 1	NPM = 123	Name = ChaneK	GPA = 4.0	Cum Laude!	<a href="#">Delete Data</a>	<a href="#">Update Data</a>
No. 2	NPM = 124	Name = ChaneK Jr	GPA = 3.0	Sangat Memuaskan	<a href="#">Delete Data</a>	<a href="#">Update Data</a>

Tutorial 07 by Disa Ainun Yolanna

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

Membuat interface CourseDAO pada dao

```

1 package com.example.dao;
2
3 import com.example.model.CourseModel;
4 import org.springframework.stereotype.Service;
5
6 import java.util.List;
7
8 @Service
9 public interface CourseDAO {
10
11     CourseModel selectCourse(String id_course);
12     List<CourseModel> selectAllCourses();
13
14 }
15

```

*Method* diatas akan diimplementasikan di CourseDAOImpl seperti sebagai berikut

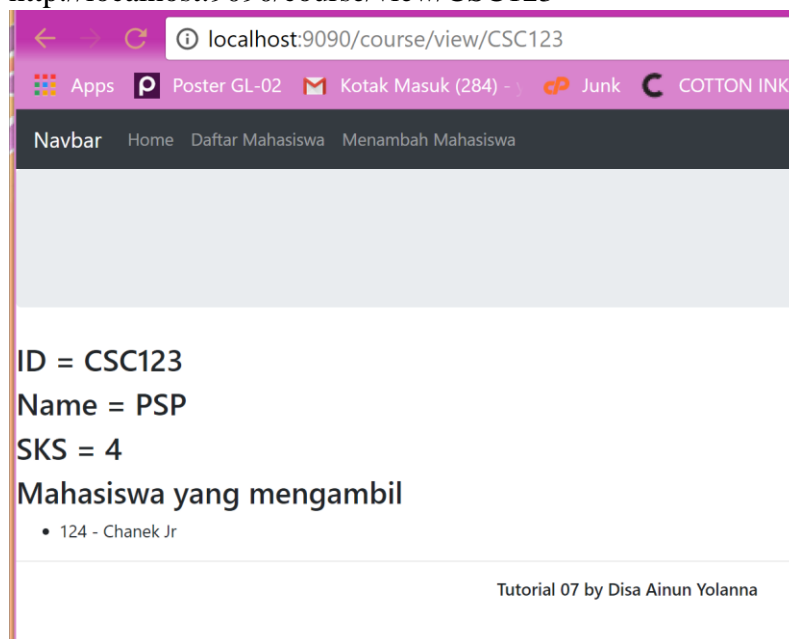
```
1 package com.example.dao;
2
3
4 import com.example.model.CourseModel;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7 import org.springframework.web.client.RestTemplate;
8
9 import java.util.List;
10
11 @Service
12 public class CourseDAOImpl implements CourseDAO {
13     @Autowired
14     private RestTemplate restTemplate;
15
16     @Override
17     public CourseModel selectCourse (String id_course){
18         CourseModel course = restTemplate.getForObject (url: "http://localhost:8080/rest/course/view/"+ id_course,
19             CourseModel.class);
20         return course;
21     }
22
23     @Override
24     public List<CourseModel> selectAllCourses () {
25         List<CourseModel> courses = restTemplate.getForObject (url: "http://localhost:8080/rest/course/viewall", List.class);
26         return courses;
27     }
28 }
29
```

Untuk *method* `selectCourse` akan menampilkan course berdasarkan id yang terdapat dalam path. *Method* tersebut akan mengembalikan *object* course tanpa mengakses database. Hal ini terjadi, karena penggunaan kelas `RestTemplate` adalah untuk mengkonsumsi objek REST web service dengan menggunakan `getForObject` yang akan menerima parameter berupa url producer web service yaitu “`http://localhost:8080/rest/course/view/{id_course}`” dan mendapat *object* course berdasarkan id. *Method* `selectCourse` pada `CourseDAO` akan dipanggil oleh class `CourseServiceRest` yang mengimplement `CourseService`.

Kemudian *method* `selectAllCourses` akan menampilkan semua course yang terdapat dalam database. *Method* tersebut akan mengembalikan *list of object* course tanpa mengakses database. Hal ini terjadi, karena penggunaan kelas `RestTemplate` adalah untuk mengkonsumsi objek REST web service dengan menggunakan `getForObject` yang akan menerima parameter berupa url producer web service yaitu “`http://localhost:8080/rest/course/viewall`” dan mendapat *list of object* courses. *Method* `selectAllCourses` pada `CouseDAO` juga akan dipanggil oleh class `CourseServiceRest` yang mengimplement `CourseService`.

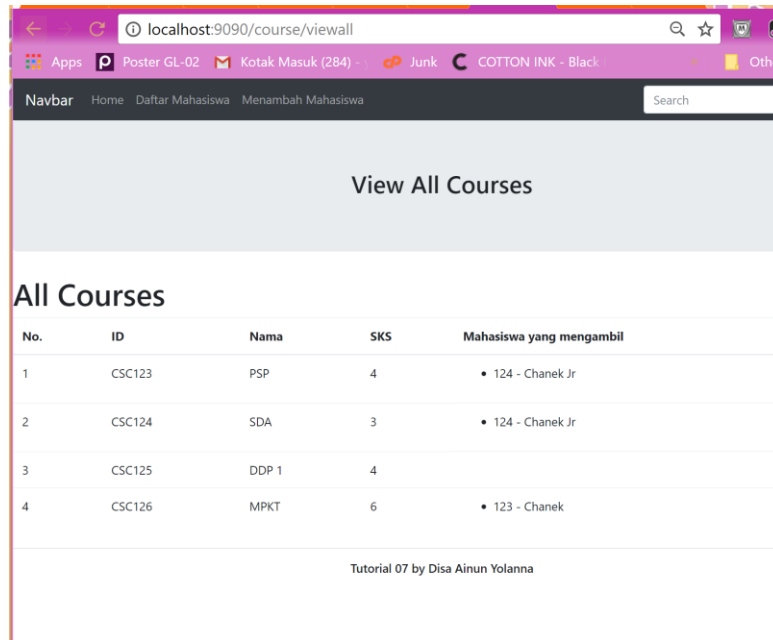
```
1 package com.example.service;
2
3 import ...
4
5 @Slf4j
6 @Service
7 @Primary
8 public class CourseServiceRest implements CourseService {
9
10     @Autowired
11     private CourseDAO courseDAO;
12
13     @Override
14     public CourseModel selectCourse(String id_course){
15         log.info("REST - select course with id {}", id_course);
16         return courseDAO.selectCourse(id_course);
17     }
18
19     @Override
20     public List<CourseModel> selectAllCourses(){
21         log.info("REST - select all courses");
22         return courseDAO.selectAllCourses();
23     }
24 }
25
26 }
```

Tampilan yang dihasilkan ketika user mengakses  
<http://localhost:9090/course/view/CSC123>



Tampilan yang dihasilkan ketika user mengakses  
<http://localhost:9090/course/viewall>





The screenshot shows a web browser at localhost:9090/course/viewall. The page has a purple header with a search bar and navigation links. Below the header is a grey button labeled 'View All Courses'. The main content area is titled 'All Courses' and contains a table with 5 columns: No., ID, Nama, SKS, and Mahasiswa yang mengambil. The table lists 4 courses. At the bottom, there is a footer that says 'Tutorial 07 by Disa Ainun Yolanna'.

No.	ID	Nama	SKS	Mahasiswa yang mengambil
1	CSC123	PSP	4	• 124 - Chanek Jr
2	CSC124	SDA	3	• 124 - Chanek Jr
3	CSC125	DDP 1	4	
4	CSC126	MPKT	6	• 123 - Chanek

Tutorial 07 by Disa Ainun Yolanna

### Hal yang dipelajari

Saya mempelajari cara membuat aplikasi service producer dan service consumer. Yang mana service producer akan menyediakan data untuk service consumer. Data tersebut dalam bentuk format JSON. Service consumer akan menerima dan menggunakan data tersebut dan akan menampilkannya, yang mana service consumer berperan menjadi frontend.