

**Tutorial 07**  
**Web Service Menggunakan Spring Boot Framework**  
Savira Nurul Ahila | 1506689692

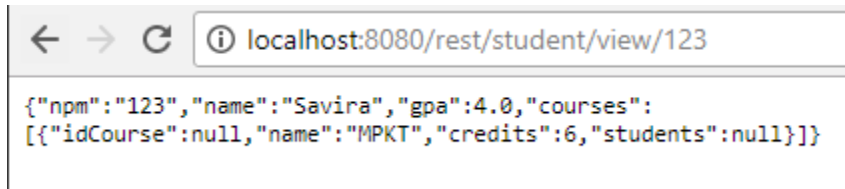
**Tutorial**

- Membuat class StudentRestController

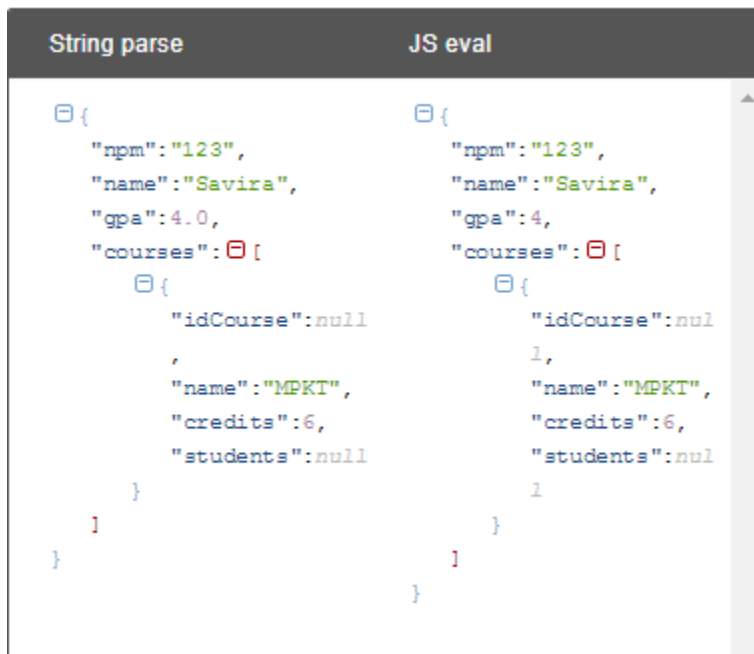
```
@RestController
@RequestMapping("/rest")
public class StudentRestController
{
    @Autowired
    StudentService studentService;

    @RequestMapping("/student/view/{npm}")
    public StudentModel view (@PathVariable(value = "npm") String npm) {
        StudentModel student = studentService.selectStudent (npm);
        return student;
    }
}
```

- buka halaman "localhost:8080/rest/student/view/123"



- Untuk memudahkan pembacaan silakan gunakan JSON View



### 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”

### Jawab:

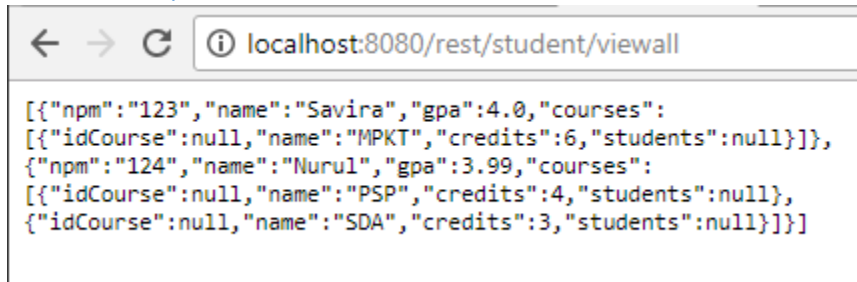
- Menambahkan kode seperti berikut di dalam class StudentRestController

```
@RequestMapping("/student/viewall")
public List<StudentModel> viewAll(){

    List<StudentModel> allStudents = studentService.selectAllStudents();
    return allStudents;

}
```

- Membuka <http://localhost:8080/rest/student/viewall>



```
[{"npm": "123", "name": "Savira", "gpa": 4.0, "courses": [{"idCourse": null, "name": "MPKT", "credits": 6, "students": null}]}, {"npm": "124", "name": "Nurul", "gpa": 3.99, "courses": [{"idCourse": null, "name": "PSP", "credits": 4, "students": null}, {"idCourse": null, "name": "SDA", "credits": 3, "students": null}]}
```

- Hasil setelah di-parse

```
{
  {
    "npm": "123",
    "name": "Savira",
    "gpa": 4.0,
    "courses": [
      {
        "idCourse":
          :null,
        "name": "MP
          KT",
        "credits":
          6,
        "students":
          :null
      }
    ]
  },
}
```

Savira Nurul Ahila  
1506689692  
Kelas B  
Tutorial 7 APAP

```
{
  "npm": "124",
  "name": "Nurul",
  "gpa": 3.99,
  "courses": [
    {
      "idCourse":
      :null,
      "name": "PS
      P",
      "credits":
      4,
      "students"
      :null
    },
    {
      "idCourse"
      :null,
      "name": "SD
      A",
      "credits":
      3,
      "students"
      :null
    }
  ]
}
```

## 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)

### Jawab:

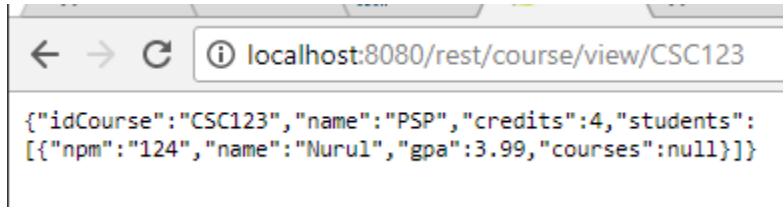
- Membuat class baru yaitu CourseRestController di dalam package rest

```
▼ com.example.rest
  > CourseRestController.java
  > StudentRestController.java
```

- Membuat kode service untuk melihat suatu course dengan masukan ID Course seperti berikut:

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

- Hasil ketika membuka <http://localhost:8080/rest/course/view/CSC123>



Hasil ketika di-parse:



- Membuat kode service untuk melihat semua course

```
@RequestMapping("/course/viewall")
public List<CourseModel> viewAllCourse(){

    List<CourseModel> allCourses = studentService.selectAllCourses();
    return allCourses;

}
```

- Menambahkan method selectAllCourses() di interface StudentService

```
List<CourseModel> selectAllCourses();
```

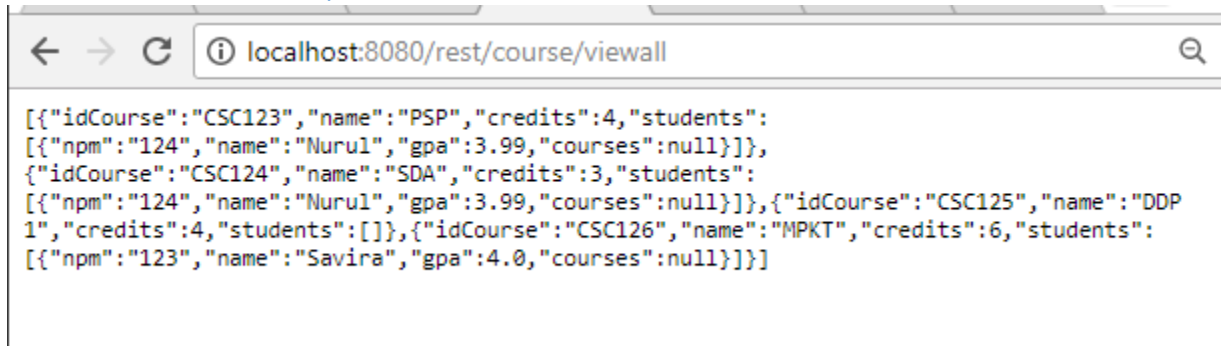
- Menambahkan method selectAllCourses() di class StudentServiceDatabase

```
@Override
public List<CourseModel> selectAllCourses(){
    return studentMapper.selectAllCourses();
}
```

- Mengimplementasikan method `selectAllCourses()` di interface `StudentMapper`

```
@Select("select id_course, name, credits from course")
@Results(value = {
    @Result(property="idCourse", 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="selectAllStudentsWithCourse"))
})
List<CourseModel> selectAllCourses();
```

- Hasil setelah membuka <http://localhost:8080/rest/course/viewall>



Savira Nurul Ahila  
1506689692  
Kelas B  
Tutorial 7 APAP

- Hasil setelah di-parse

#### String parse

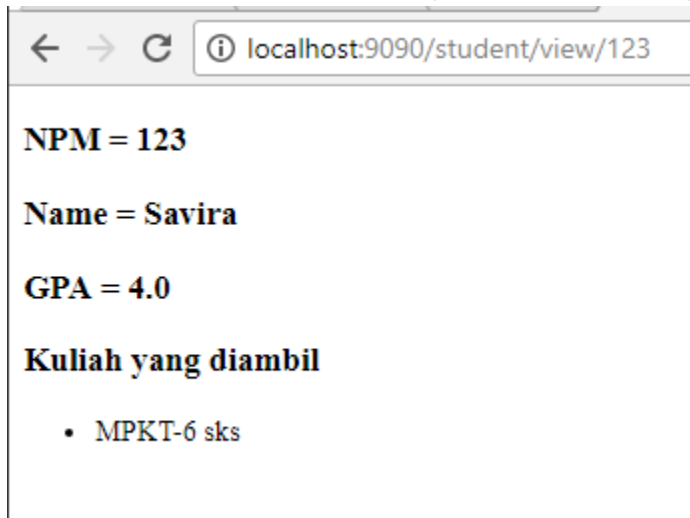
```
[
  {
    "idCourse": "CSC123",
    "name": "PSP",
    "credits": 4,
    "students": [
      {
        "npm": "124",
        "name": "Nurul",
        "gpa": 3.99,
        "courses": null
      }
    ]
  },
  {
    "idCourse": "CSC124",
    "name": "SDA",
    "credits": 3,
    "students": [
      {
        "npm": "124",
        "name": "Nurul",
        "gpa": 3.99,
        "courses": null
      }
    ]
  }
],
```

Savira Nurul Ahila  
1506689692  
Kelas B  
Tutorial 7 APAP

```
..
  {
    "idCourse": "CSC125",
    "name": "DDP 1",
    "credits": 4,
    "students": [
    ],
  },
  {
    "idCourse": "CSC126",
    "name": "MPKT",
    "credits": 6,
    "students": [
      {
        "npm": "123",
        "name": "Savira",
        "gpa": 4.0,
        "courses": null
      }
    ]
  }
]
```

## Tutorial

- Setelah membuat service consumer, buka localhost:9090/student/view/123



- Di console akan tertulis:

```
2017-11-04 19:00:31.865 INFO 11136 --- [nio-9090-exec-4] com.example.service.StudentServiceRest : REST - select student with npm 123
```

### Latihan 3:

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

#### Jawab:

- Mencantumkan method selectAllStudents() pada interface StudentDAO

```
public interface StudentDAO {  
  
    StudentModel selectStudent (String npm);  
    List<StudentModel> selectAllStudents ();  
  
}
```

- Mengimplementasikan method selectAllStudents() pada class StudentDAOImpl dengan kode sebagai berikut:

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

- Memanggil method selectAllStudents melalui class StudentServiceRest menggunakan kode seperti berikut

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

- Hasil ketika membuka <http://localhost:9090/student/viewall>

#### All Students

Show 10 ▾ entries					Search: <input type="text"/>	
No	NPM	Name	GPA	Cum laude	Delete	Update
No. 1	NPM = 123	Name = Savira	GPA = 4.0	Cum Laude!	<button>Delete Data</button>	<button>Update Data</button>
No. 2	NPM = 124	Name = Nurul	GPA = 3.99	Cum Laude!	<button>Delete Data</button>	<button>Update Data</button>
Showing 1 to 2 of 2 entries					Previous	1 Next
Mata Kuliah APAP						

### Latihan 4:

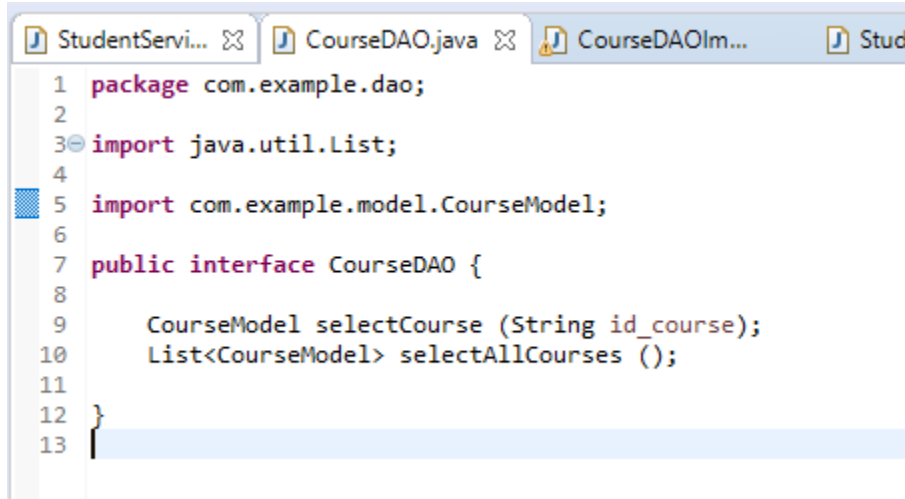
Implementasikan service consumer untuk class CourseModel dengan membuat class-class DAO dan service baru



Savira Nurul Ahila  
1506689692  
Kelas B  
Tutorial 7 APAP

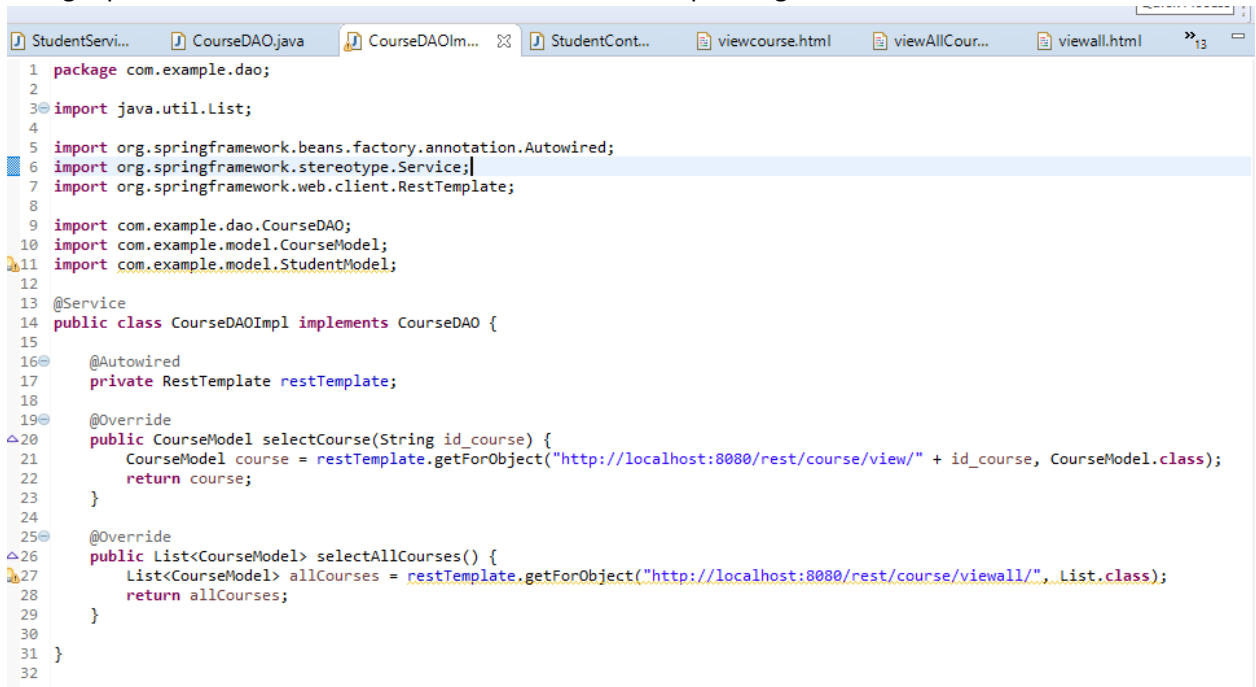
Jawab:

- Mendefinisikan method apa saja yang dapat dilakukan di dalam file courseDAO



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

- Mengimplementasikan method tersebut di courseDAOImpl sebagai berikut



```
1 package com.example.dao;
2
3 import java.util.List;
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.dao.CourseDAO;
10 import com.example.model.CourseModel;
11 import com.example.model.StudentModel;
12
13 @Service
14 public class CourseDAOImpl implements CourseDAO {
15
16     @Autowired
17     private RestTemplate restTemplate;
18
19     @Override
20     public CourseModel selectCourse(String id_course) {
21         CourseModel course = restTemplate.getForObject("http://localhost:8080/rest/course/view/" + id_course, CourseModel.class);
22         return course;
23     }
24
25     @Override
26     public List<CourseModel> selectAllCourses() {
27         List<CourseModel> allCourses = restTemplate.getForObject("http://localhost:8080/rest/course/viewall/", List.class);
28         return allCourses;
29     }
30
31 }
32
```

- Memanggil method tersebut melalui StudentServiceRest

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

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

- Menambahkan controller untuk viewAll di StudentController

```
@RequestMapping("/course/viewall")
public String viewAllCourses(Model model) {
    List<CourseModel> allCourses = studentDAO.selectAllCourses();

    if(allCourses != null) {
        model.addAttribute("allCourses", allCourses);
        return "viewAllCourses";
    } else {
        model.addAttribute("allCourses", allCourses);
        return "not-found";
    }
}
```

- Menambahkan html viewAllCourses

```
27     $('#view-data').DataTable();
28     });
29 </script>
30 <title>View All Courses</title>
31 </head>
32 <body>
33
34     <div th:replace="fragments/fragment :: header"></div>
35     <h1>All Students</h1>
36     <table id="view-data" class="display" cellspacing="0" width="100%">
37         <thead>
38             <tr>
39                 <th>No</th>
40                 <th>ID</th>
41                 <th>Name</th>
42                 <th>Credits</th>
43             </tr>
44         </thead>
45         <tbody>
46             <div th:each="allCourses, iterationStatus: ${allCourses}"
47                 th:class="${iterationStatus.odd}? 'odd'">
48                 <tr>
49                     <td><h3 th:text="${iterationStatus.count}"></h3></td>
50                     <td><h3 th:text="${allCourses.idCourse}"></h3></td>
51                     <td><h3 th:text="${allCourses.name}"></h3></td>
52                     <td><h3 th:text="${allCourses.credits}"></h3></td>
53                 </tr>
54             </div>
55         </tbody>
56     </table>
57     <div th:replace="fragments/fragment :: footer"></div>
58 </body>
59 </html>
60
```

#### Hal yang dipelajari dari Tutorial07:

Melalui tutorial ke-7 ini saya mempelajari tentang pemisahan layer dalam suatu program, yaitu layer untuk consumer dan layer untuk producer. Hal ini dibuat agar layer consumer tidak perlu mengakses ke database, sebagai gantinya layer producer lah yang akan memberikan data JSON ke layer consumer. Di sini layer consumer berperan sebagai front-end sedangkan layer producer berperan sebagai back-end.