

## WRITE UP TUTORIAL 7

### Membuat Web Service Menggunakan Spring Boot Framework

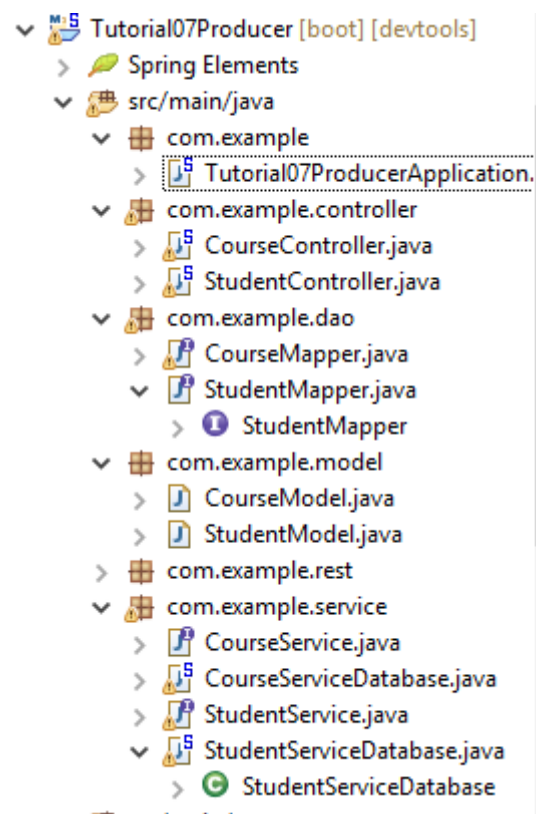
#### Persiapan

Membuat Folder Tutorial07 dengan isi Tutorial07Consumer dan Tutorial07Producer

 .git	04/11/2017 09.26	File folder
 Tutorial07Consumer	04/11/2017 08.33	File folder
 Tutorial07Producer	04/11/2017 00.40	File folder


#### Membuat Service Producer

Melakukan Import Tutorial05 ke dalam Tutorial07Producer



(Rest sudah dibuat karena writeup dibuat setelah melakukan tutorial)

Lalu buat package baru com.example.rest seperti berikut:

>  com.example.rest

Di dalamnya, dibuat StudentRestController.java

## Tutorial 07

Mohammad Izzat Raihan

1506689446

```
package com.example.rest;

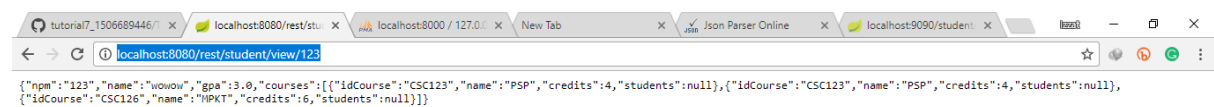
import java.util.List;

@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);
        System.out.println(student.getCourses().get(0).getIdCourse());
        return student;
    }
}
```

Setelah dibuat, kita dapat melihat hasilnya langsung melalui localhost:8080/rest/student/view/123



Jika dilihat pada Json Parse seperti berikut



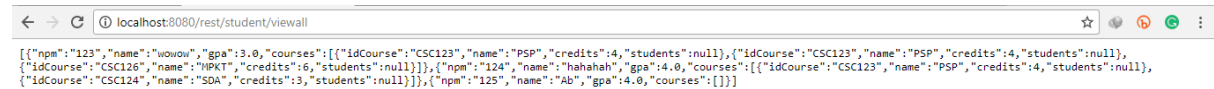
## Latihan 1

Buat 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"

1. Buatlah method pada Rest Controller

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

Lalu tampilkan akan seperti berikut



```
[{"npm": "123", "name": "wowow", "gpa": 3.0, "courses": [{"idCourse": "CSC123", "name": "PSP", "credits": 4, "students": null}, {"idCourse": "CSC126", "name": "MPKT", "credits": 6, "students": null}]}, {"npm": "124", "name": "hahahah", "gpa": 4.0, "courses": [{"idCourse": "CSC123", "name": "PSP", "credits": 4, "students": null}]}, {"npm": "125", "name": "Ab", "gpa": 4.0, "courses": []}]
```

Setelah diparse:

String parse	JS eval
<pre>[{"npm": "123", "name": "wowow", "gpa": 3.0, "courses": [{"idCourse": "CSC123", "name": "PSP", "credits": 4, "students": null}, {"idCourse": "CSC126", "name": "MPKT", "credits": 6, "students": null}]}, {"npm": "124", "name": "hahahah", "gpa": 4.0, "courses": [{"idCourse": "CSC123", "name": "PSP", "credits": 4, "students": null}]}]</pre>	<pre>[{"npm": "123", "name": "wowow", "gpa": 3, "courses": [{"idCourse": "CSC123", "name": "PSP", "credits": 4, "students": null}, {"idCourse": "CSC126", "name": "MPKT", "credits": 6, "students": null}]}, {"npm": "124", "name": "hahahah", "gpa": 4, "courses": [{"idCourse": "CSC123", "name": "PSP", "credits": 4, "students": null}]}]</pre>

## 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).

1. Buat service untuk class Course:

```
com.example.service
├── CourseService.java
└── CourseServiceDatabase.java
```

```
public interface CourseService
{
    CourseModel selectCourse (String id);

    List<CourseModel> selectAllCourse ();
}

package com.example.service;

import java.util.List;

@Slf4j
@Service
public class CourseServiceDatabase implements CourseService
{
    @Autowired
    private CourseMapper courseMapper;

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

    @Override
    public List<CourseModel> selectAllCourse() {
        log.info ("select all course");
        return courseMapper.selectAllCourse();
    }
}
```

## 2. Buat Rest Controller untuk Course

```
package com.example.rest;

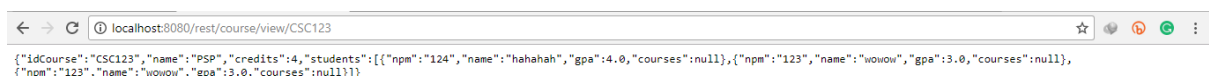
import java.util.List;

@RestController
@RequestMapping("/rest")
public class CourseRestController {
    @Autowired
    CourseService courseService;

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

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

Lalu lihat hasilnya:



The screenshot shows a web browser window with the address bar displaying `localhost:8080/rest/course/view/CSC123`. The response body is a JSON array containing two objects:

```
{
  "idCourse": "CSC123", "name": "PSP", "credits": 4, "students": [
    {
      "npm": "124", "name": "hahahah", "gpa": 4.0, "courses": null
    },
    {
      "npm": "123", "name": "wowow", "gpa": 3.0, "courses": null
    }
  ]
},
{
  "npm": "123", "name": "wowow", "gpa": 3.0, "courses": null
}
```

Tutorial 07  
Mohammad Izzat Raihan  
1506689446  
Setelah diparse:

String parse	JS eval
<pre>{   "idCourse": "CSC123",   "name": "PSP",   "credits": 4,   "students": [     {       "npm": "124",       "name": "hahahah",       "gpa": 4.0,       "courses": null     },     {       "npm": "123",       "name": "wowow",       "gpa": 3.0,       "courses": null     },     {       "npm": "123",       "name": "wowow",       "gpa": 3.0,       "courses": null     }   ] }</pre>	<pre>{   "idCourse": "CSC123",   "name": "PSP",   "credits": 4,   "students": [     {       "npm": "124",       "name": "hahahah",       "gpa": 4,       "courses": null     },     {       "npm": "123",       "name": "wowow",       "gpa": 3,       "courses": null     },     {       "npm": "123",       "name": "wowow",       "gpa": 3,       "courses": null     }   ] }</pre>

Lalu untuk viewall:

```
localhost:8080/rest/course/viewall
[{"idCourse": "CSC123", "name": "PSP", "credits": 4, "students": [{"npm": "124", "name": "hahahah", "gpa": 4.0, "courses": null}, {"npm": "123", "name": "wowow", "gpa": 3.0, "courses": null}, {"npm": "123", "name": "wowow", "gpa": 3.0, "courses": null}], {"idCourse": "CSC124", "name": "SDA", "credits": 3, "students": [{"npm": "124", "name": "hahahah", "gpa": 4.0, "courses": null}], {"idCourse": "CSC125", "name": "DDP 1", "credits": 4, "students": []}, {"idCourse": "CSC126", "name": "MPKT", "credits": 6, "students": [{"npm": "123", "name": "wowow", "gpa": 3.0, "courses": null}]}
```

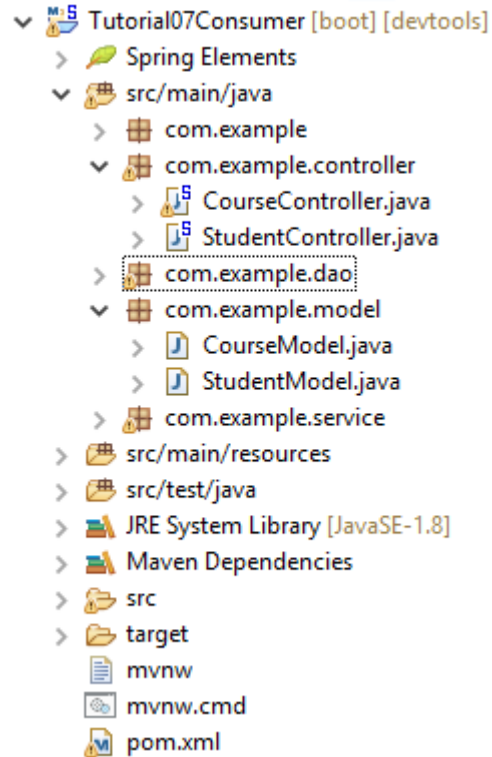
Setelah diparse:

Tutorial 07  
Mohammad Izzat Raihan  
1506689446

String parse	JS eval
<pre>[   {     "idCourse": "CSC123",     "name": "PSP",     "credits": 4,     "students": [       {         "npm": "124",         "name": "hahahah",         "gpa": 4.0,         "courses": null       },       {         "npm": "123",         "name": "wowow",         "gpa": 3.0,         "courses": null       },       {         "npm": "123",         "name": "wowow",         "gpa": 3.0,         "courses": null       }     ]   },   {     "idCourse": "CSC124",     "name": "SDA",     "credits": 3,     "students": [       {</pre>	<pre>[   {     "idCourse": "CSC123",     "name": "PSP",     "credits": 4,     "students": [       {         "npm": "124",         "name": "hahahah",         "gpa": 4,         "courses": null       },       {         "npm": "123",         "name": "wowow",         "gpa": 3,         "courses": null       },       {         "npm": "123",         "name": "wowow",         "gpa": 3,         "courses": null       }     ]   },   {     "idCourse": "CSC124",     "name": "SDA",     "credits": 3,     "students": [       {</pre>

## Membuat Service Consumer

Import tutorial 6 ke dalam Tutorial07Consumer.java



Ubah port pada application.properties menjadi 9090

```
server.port=9090
```

Buat interface StudentDAO pada DAO

```
package com.example.dao;

import java.util.List;

import com.example.model.StudentModel;

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

Buat implementasinya StudentDAOImpl

Tutorial 07  
Mohammad Izzat Raihan  
1506689446

```
@Service
public class StudentDAOImpl implements StudentDAO {
    @Autowired
    private RestTemplate restTemplate;

    /**@Bean
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }*/

    @Override
    public StudentModel selectStudent(String npm) {
        StudentModel student = restTemplate.getForObject("http://localhost:8080/rest/student/view/" + npm,
            StudentModel.class);
        return student;
    }

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

Buatlah pada service, StudentServiceRest yang mengimplementasi StudentService

```
@Slf4j
@Service
@Primary
public class StudentServiceRest implements StudentService {
    @Autowired
    private StudentDAO studentDAO;

    @Override
    public StudentModel selectStudent(String npm) {
        log.info("REST - select student with npm {}", npm);
        return studentDAO.selectStudent(npm);
    }

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

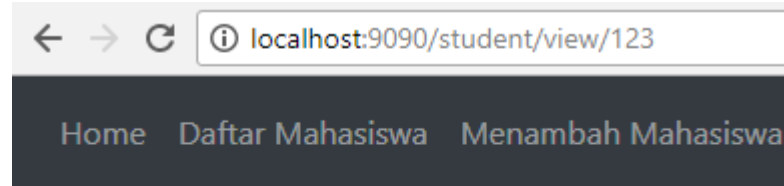
    @Override
    public void addStudent(StudentModel student) {
    }

    @Override
    public void deleteStudent(String npm) {
    }

    @Override
    public void updateStudent(StudentModel student) {
    }
}
```

Jalankan kedua Program secara bersamaan (producer dan consumer)





NPM = 123

Name = wowow

GPA = 3.0

Kuliah yang diambil

- PSP-4 sks
- PSP-4 sks
- MPKT-6 sks

```
Tutorial07Consumer - Tutorial07ConsumerApplication [Spring Boot App] C:\Program Files\Java\jdk1.8.0_73\bin\javaw.exe (4 Nov 2017 12:58:06)
2017-11-04 13:40:54.023 INFO 9652 --- [nio-9090-exec-1] 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

1. Pada StudentDAO sudah terdapat method untuk selectAllStudents lalu implementasi pada StudentDAOImpl

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

2. Pada StudentServiceRest diperbaiki menjadi

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

Lalu setelah dijalankan:

localhost:9090/student/viewall

View All Students Home Daftar Mahasiswa Menambah Mahasiswa

## All Students

Show 10 entries Search:

No.	NPM	Name	GPA	Status	Delete	Update
1	123	wowow	3.0	Sangat Memuaskan	<a href="#">Delete Data</a>	<a href="#">Update Data</a>
2	124	hahahah	4.0	Cum Laude!	<a href="#">Delete Data</a>	<a href="#">Update Data</a>
3	125	Ab	4.0	Cum Laude!	<a href="#">Delete Data</a>	<a href="#">Update Data</a>

Showing 1 to 3 of 3 entries Previous 1 Next

## Latihan 4

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

1. Buat service untuk Course dan service rest

```
> CourseService.java
> CourseServiceRest.java

public interface CourseService {
    CourseModel selectCourse (String id_course);

    List<CourseModel> selectAllCourse();
}

@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);
    }

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

2. Buat juga CourseDAO dan implementasinya

```
public interface CourseDAO {

    CourseModel selectCourse(String id);

    List<CourseModel> selectAllCourse();
}
```

```
@Service
public class CourseDAOImpl implements CourseDAO {

    @Autowired
    private RestTemplate restTemplate;

    @Bean
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }

    @Override
    public CourseModel selectCourse(String id) {
        CourseModel course = restTemplate.getForObject("http://localhost:8080/rest/course/view/" + id,
            CourseModel.class);
        return course;
    }

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

3. Implementasikan viewall course pada controller

```
@RequestMapping("/course/view/{id_course}")
public String viewPath (Model model,
    @PathVariable(value = "id_course") String id_course)
{
    CourseModel course = courseDAO.selectCourse (id_course);

    if (course != null) {
        model.addAttribute ("course", course);
        return "view-course";
    } else {
        model.addAttribute ("id_course", id_course);
        return "not-found-course";
    }
}

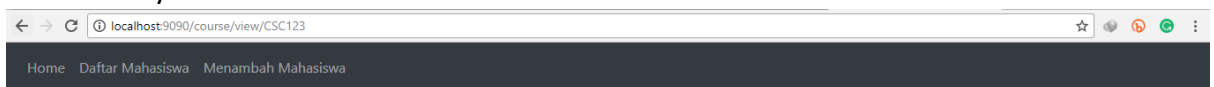
@RequestMapping("/course/viewall")
public String view (Model model)
{
    List<CourseModel> courses = courseDAO.selectAllCourse();
    model.addAttribute ("courses", courses);

    return "viewall-course";
}
```

4. Tidak lupa untuk membuat html untuk viewall-coursenya

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <title th:replace="fragments/fragment :: title"></title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
    <link rel="stylesheet" type="text/css" href="/css/datatables.min.css"/>
    <link rel="stylesheet" type="text/css" href="/css/bootstrap.min.css"/>
    <script type="text/javascript" src="/js/bootstrap.min.js"></script>
  </head>
  <body>
    <div th:replace="fragments/fragment :: header"></div>
    <h1>All Courses</h1>
    <table id="allStudents" class="display" style="text-align:center;">
      <thead>
        <tr>
          <th>No.</th>
          <th>ID</th>
          <th>Name</th>
          <th>Credits</th>
        </tr>
      </thead>
      <tbody>
        <tr th:each="course, iterationStatus: ${courses}" th:class="${iterationStatus.odd}?'odd'">
          <td th:text="${iterationStatus.count}">No. 1</td>
          <td th:text="${course.idCourse}">Course ID</td>
          <td th:text="${course.name}">Course Name</td>
          <td th:text="${course.credits}">Course Credits</td>
        </tr>
      </tbody>
    </table>
    <script type="text/javascript" src="/js/datatables.min.js"></script>
  </body>
</html>
```

## 5. Lihat hasilnya:



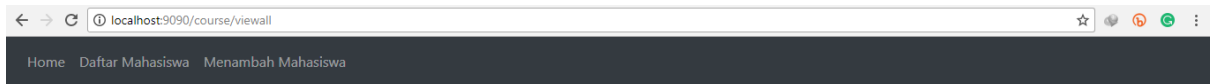
id\_course = CSC123

Name = PSP

Credits = 4

Mahasiswa yang mengambil

- 124-hahahah
- 123-wowow
- 123-wowow



## All Courses

Show  entries

Search:

No.	ID	Name	Credits
1	CSC123	PSP	4
2	CSC124	SDA	3
3	CSC125	DDP 1	4
4	CSC126	MPKT	6

Showing 1 to 4 of 4 entries

Previous  Next

## **Lesson Learned**

Pada tutorial kali ini, saya mempelajari bagaimana cara menggunakan web service pada SpringBoot Framework. Untuk menjalankannya, diperlukan dua program yaitu Producer yang berfungsi untuk mengakses data langsung dari database dan mereturn objek masing-masing serta Consumer yang berfungsi untuk “menampilkan” fungsi tanpa perlu mengakses data secara langsung dari database dengan bantuan Producer dan REST.

1. Method pada latihan 1 viewall mirip dengan yang dicontohkan pada tutorial, hanya saja, yang diambil adalah dalam bentuk `list<StudentModel>`
2. Method pada latihan 2 hanya berubah pada bagian Course (menggantikan Student)

Pada Consumer, diperlukan DAO baru (interface dan implementasinya) untuk mengakses data melalui perantara Producer (tidak langsung dari database). Hal ini kemudian diakses melalui Service dengan implementasi ServiceRest dengan anotasi `@Primary` agar implementasi yang dilakukan adalah yang ada pada ServiceRest dan bukan pada ServiceDatabase.

3. Method yang digunakan pada latihan 3 mirip dengan yang dicontohkan pada tutorial, hanya saja yang diambil adalah `list<StudentModel>` untuk mendapatkan seluruh Students.
4. Method yang digunakan pada latihan 4 hanya merubah Student dan Coursenya saja.