Enrico Jano Fiandi
1506689490
Tutorial 7 ADPAP-C

Latihan 1

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

Method viewall() akan dieksekusi ketika *end point* /rest/student/viewall dipanggil melalui URL, method tersebut akan mengembalikan semua data student yang terdapat di dalam database dalam bentuk JSON, berikut adalah output ketika *end point* tersebut diakses


```json
[
    {
        "npm": "123",
        "name": "enrico jano fiandi",
        "gpa": 3.5,
        "courses": [
            {
                "idCourse": "CSC126",
                "name": "MPKT",
                "credits": 6,
                "students": null
            }
        ]
    },
    {
        "npm": "124",
        "name": "hehehuhu",
        "gpa": 3.4,
        "courses": [
            {
                "idCourse": "CSC123",
                "name": "PSP",
                "credits": 4,
                "students": null
            },
            {
                "idCourse": "CSC124",
                "name": "SDA",
                "credits": 3,
                "students": null
            }
        ]
    }
]
```

Latihan 2

```java
package com.example.rest;

import java.util.List;

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

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

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

Terdapat 2 buah *end point* di dalam CourseRestController yaitu /rest/course/view/{id_course} dan /rest/course/viewall. *End point* /rest/course/view/{id_course} ketika diakses melalui url akan mengembalikan data course berdasarkan id_course yang terdapat pada url dan *end point* /rest/course/viewall akan mengembalikan semua data course di dalam database. Kedua *end point* tersebut akan mengembalikan data dalam bentuk JSON

Berikut adalah hasil ketika *endpoint* /rest/course/view/CSC123 diakses


```json
{
    "idCourse": "CSC123",
    "name": "PSP",
    "credits": 4,
    "students": [
        {
            "npm": "124",
            "name": "hehehuhu",
            "gpa": 3.4,
            "courses": null
        }
    ]
}
```

Berikut adalah hasil ketika *end point* /rest/course/viewall diakses


```
[
    {
        "idCourse": "CSC123",
        "name": "PSP",
        "credits": 4,
        "students": [
            {
                "npm": "124",
                "name": "hehehuhu",
                "gpa": 3.4,
                "courses": null
            }
        ]
    },
    {
        "idCourse": "CSC124",
        "name": "SDA",
        "credits": 3,
        "students": [
            {
                "npm": "124",
                "name": "hehehuhu",
                "gpa": 3.4,
                "courses": null
            }
        ]
    },
    {
        "idCourse": "CSC125",
        "name": "DDP 1",
        "credits": 4,
        "students": []
    },
    {
        "idCourse": "CSC126",
        "name": "MPKT",
        "credits": 6
```

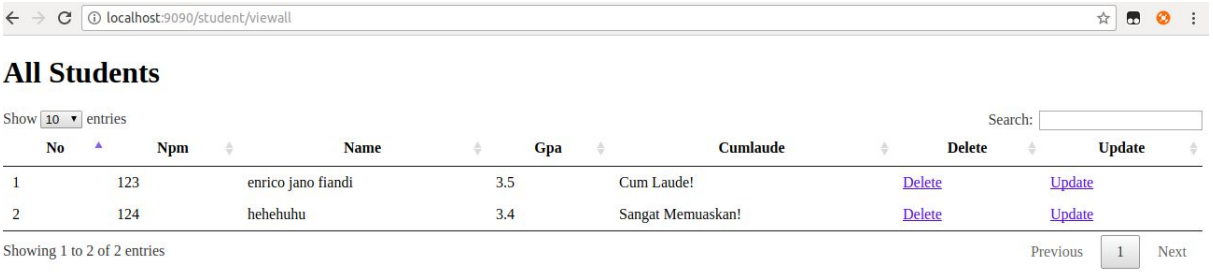

Query selectCourse sudah telah diimplementaskan pada tutorial sebelumnya, berikut adalah query selectAllCourses yang baru diimplementasikan untuk menampilkan semua data course yang ada di dalam database pada tutorial ini

```
@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="selectParticipants"))
    })
List<CourseModel> selectAllCourses();
```

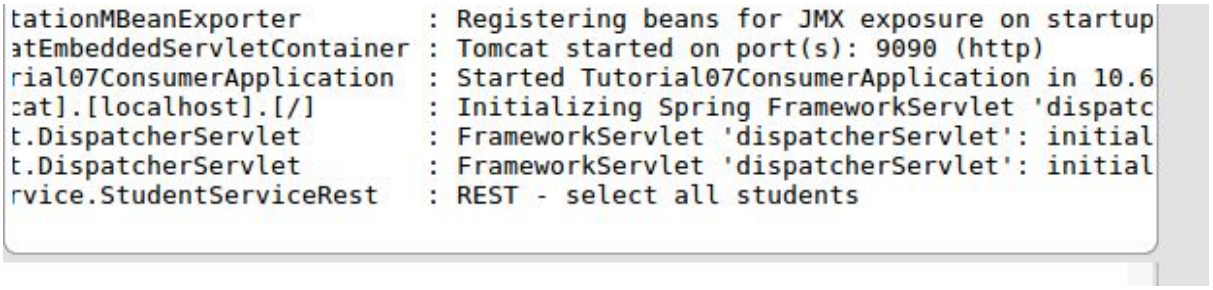
Latihan 3

```
@Override
public List<StudentModel> selectAllStudents() {
    ResponseEntity<List<StudentModel>> rateResponse =
            restTemplate.exchange("http://localhost:8080/rest/student/viewall",
                    HttpMethod.GET, null, new ParameterizedTypeReference<List<StudentModel>>() {
                });
    List<StudentModel> students = rateResponse.getBody();
    return students;
}
```

Method diatas berfungsi untuk memanggil *end point* /rest/student/viewall dalam bentuk JSON dan di convert menjadi List of StudentModel yang nantinya akan digunakan untuk menampilkan data semua student dalam database pada viewall.html. Saya menggunakan restTemplate.exchange() karena pemanggilan json dalam bentuk list tidak bisa dilakukan dengan restTemplate.getForObject(). Ketika dipanggil list tersebut akan disimpan pada body ResponseEntity.



Berikut adalah bukti dimana data semua student tersebut dipanggil melalui StudentServiceRest




```
}

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

Latihan 4

```java
package com.example.dao;

import java.util.List;

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

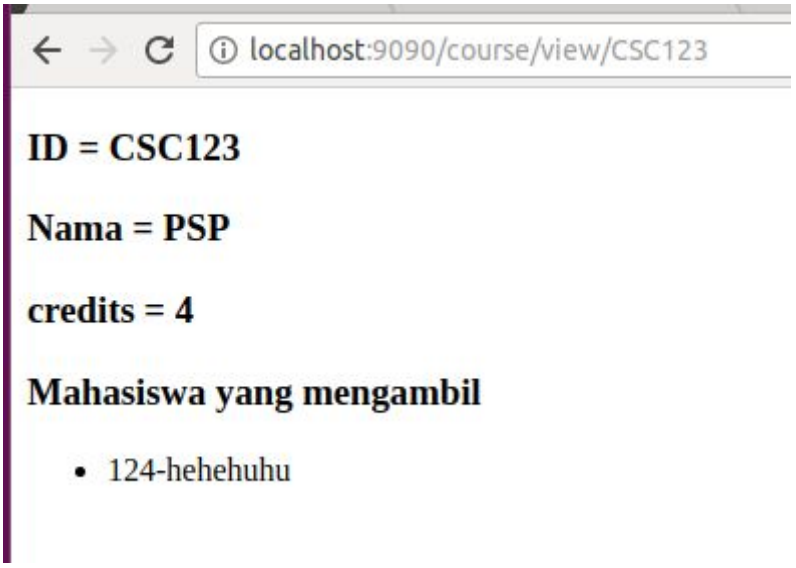
Interface CourseDAO berfungsi untuk menyiapkan method yang akan diimplementasikan oleh class CourseDAOImpl1 seperti method selectCourse dan method selectAllCourse

```java
package com.example.dao;

import java.util.List;

@Service
public class CourseDAOimp1 implements CourseDAO {

    @Autowired
    private RestTemplate restTemplate;

    @Override
    public CourseModel selectCourse(String id_course) {
        // TODO Auto-generated method stub
        CourseModel course = restTemplate.getForObject("http://localhost:8080/rest/course/view/"+id_course, CourseModel.class);
        return course;
    }

    @Override
    public List<CourseModel> selectAllCourses() {
        // TODO Auto-generated method stub
        ResponseEntity<List<CourseModel>> rateResponse =
                restTemplate.exchange("http://localhost:8080/rest/course/viewall",
                        HttpMethod.GET, null, new ParameterizedTypeReference<List<CourseModel>>() {
                    });
        List<CourseModel> courses = rateResponse.getBody();
        return courses;
    }

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

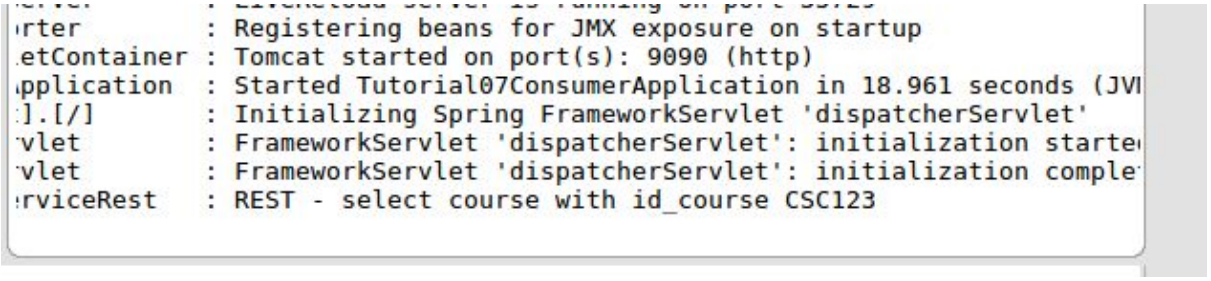
Pada class CourseDAimp1 terdapat 3 buat method yaitu selectCourse, selectAllCourse, dan restTemplate()

Penjelasan method selectCourse()

Method tersebut berfungsi untuk memanggil *end point* /rest/course/view/{id_course} dari producer yang nantinya akan mengembalikan data sebuah course berdasarkan id_course dalam bentuk JSON yang nantinya akan diconvert menjadi sebuah objek CourseModel. Objek tersebut akan digunakan untuk menampilkan data course pada *end point* /course/view/{id_course} pada consumer

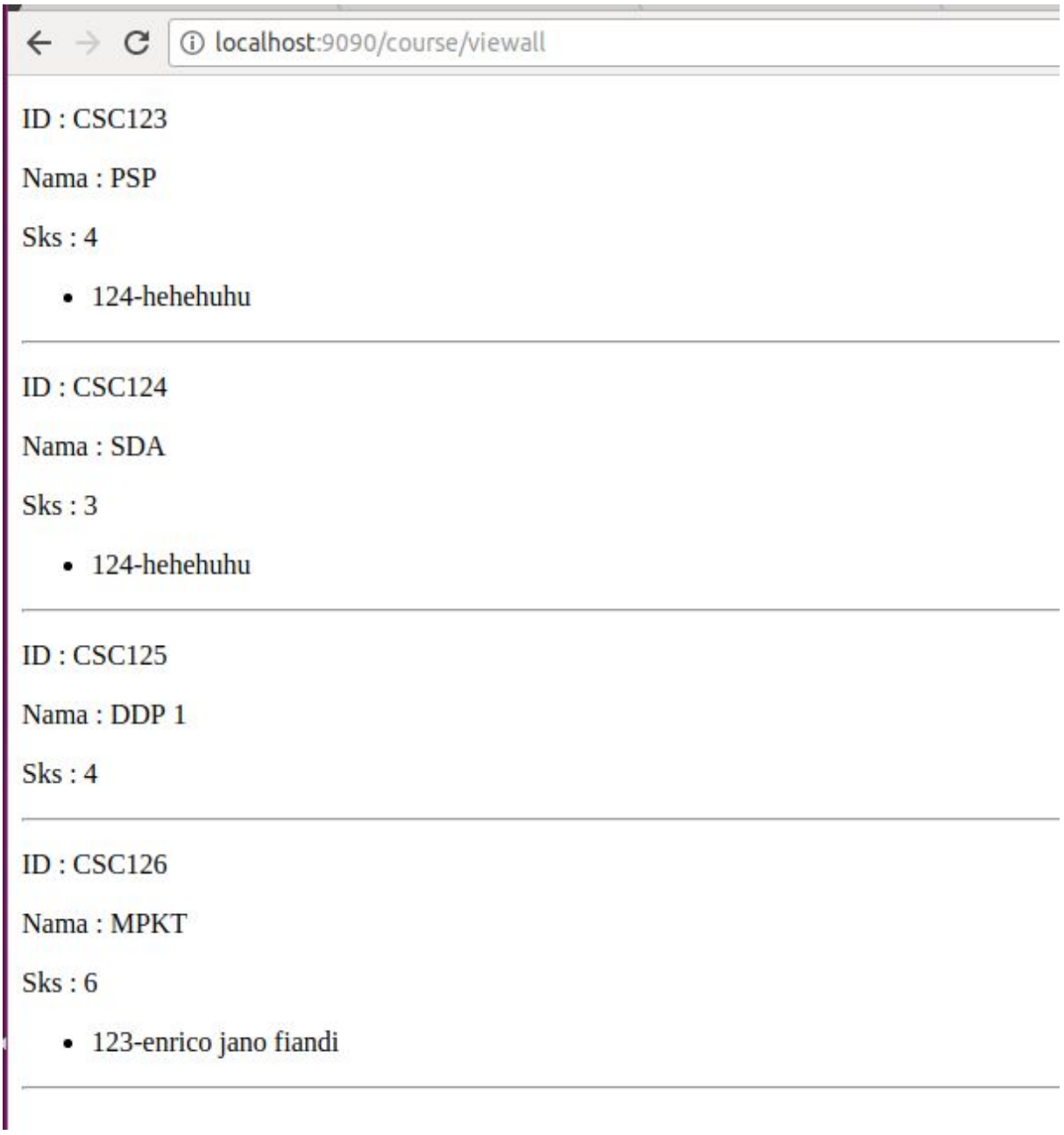


Bukti jika data diakses melalui REST



Penjelasan method selectAllCourse()

Method tersebut berfungsi untuk memanggil *end point* /rest/course/viewall dari producer yang nantinya akan mengembalikan semua data course di dalam database dalam bentuk JSON yang nantinya akan di convert menjadi sebuah List of CourseModel, saya menggunakan exhange() karena getForObject() hanya dapat men-convert JSON menjadi objek, sedangkan untuk mengconvert JSON menjadi sebuah list harus menggunakan exhange(). List of CourseModel tersebut akan digunakan untuk menampilkan data semua course pada *end point* /course/viewall pada consumer



Bukti jika data diakses melalui REST

```
al07ConsumerApplication  : Started Tutorial07ConsumerApplication in 8.992
t].[localhost].[/]       : Initializing Spring FrameworkServlet 'dispatch
DispatcherServlet        : FrameworkServlet 'dispatcherServlet': initiali
DispatcherServlet        : FrameworkServlet 'dispatcherServlet': initiali
ice.StudentServiceRest   : REST - select all courses
```

Penjelasan restTemplate()

Dikarenakan bean tidak ditemukan saat proses @Autowired dilakukan untuk variale restTemplate maka perlu dibuat methodnya.

Hal yang dipelajari dalam tutorial ini

Pada tutorial ini saya belajar bagaimana dua buah project dapat berinteraksi satu sama lain menggunakan konsep web service yang dimana salah satu project akan bertindak sebagai producer yang dimana bertugas untuk menjadi *supplier* dari data dan akan mengirimkan ke project lainnya dalam bentuk JSON.