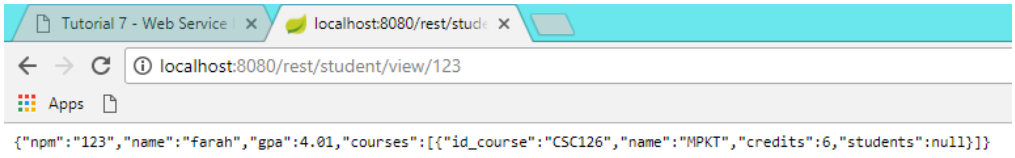
Farah Agia Ramadhina
1506731580
APAP - A

Tutorial

Membuat Rest untuk View Student by NPM

```
@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;
    }
}
```

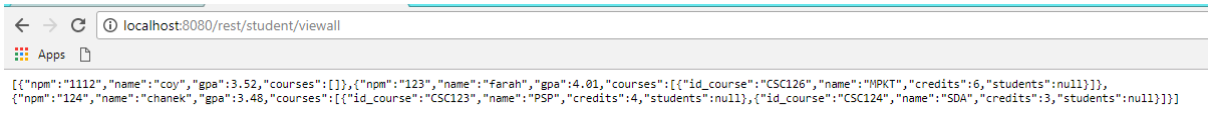


- ⇨ Perbedaan antara fungsi Controller dan Rest adalah pada rest mengembalikan objek, sedangkan Controller mengembalikan String

Latihan

- Producer
    1. **Membuat Rest Pada viewAll Student**

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

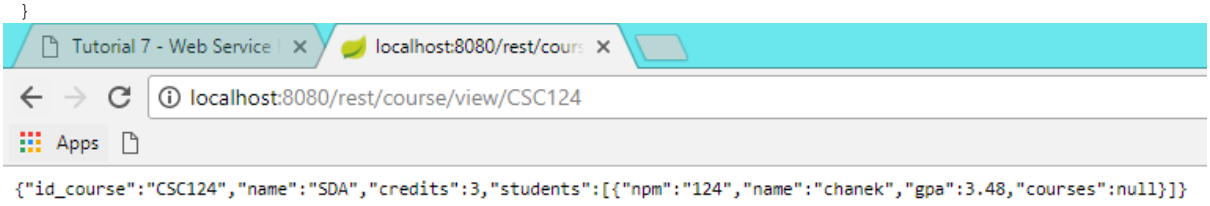


Farah Agia Ramadhina
1506731580
APAP - A

Pada kali ini sudah terlihat bahwa yang diparse adalah objek. Seluruh student yang ada di database ditampilkan di halaman. Sama halnya seperti yang ada di controller, maka akan dipetakan dengan menambah alamay “rest/” diikuti “student/viewall”

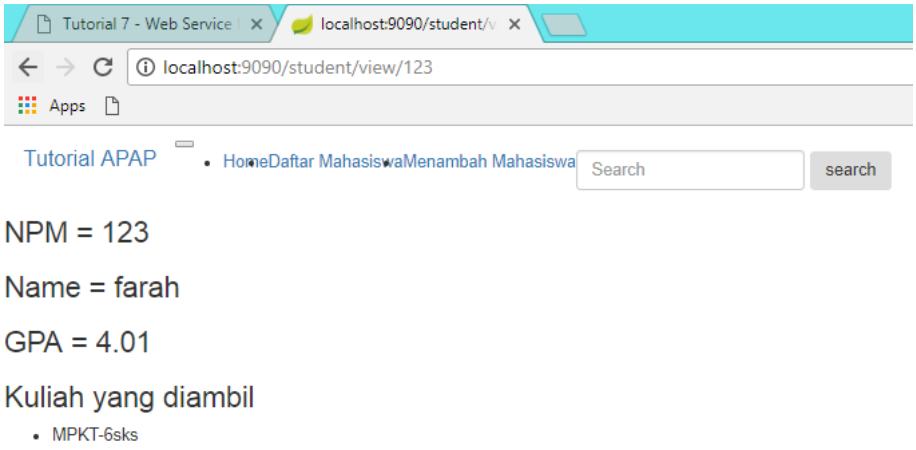
2. Membuat Rest pada View Course by Id

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



Sama seperti konsep yang sebelumnya bahwa yang diparse adalah objek yakni Course. Seluruh Course yang ada di database ditampilkan di halaman. Sama halnya seperti yang ada di controller, maka akan dipetakan dengan menambah alamat “rest/” diikuti “student/view/{id_course}” ke masing-masing id

- Consumer



Dengan cara

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

Farah Agia Ramadhina
1506731580
APAP - A

```
@Service
public class StudentDAOImpl implements StudentDAO
{
      @Autowired
      private RestTemplate 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 ()
    {
          return null;
    }
```

⇨ Berbeda dengan sebelumnya, di consumer ini digunakan getForObject yang bukan hanya memparsing object tapi langsung ke URL nya

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

StudentDAOImpls

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

⇨ Befungsi memparsing Object ke Url

StudentDAO

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

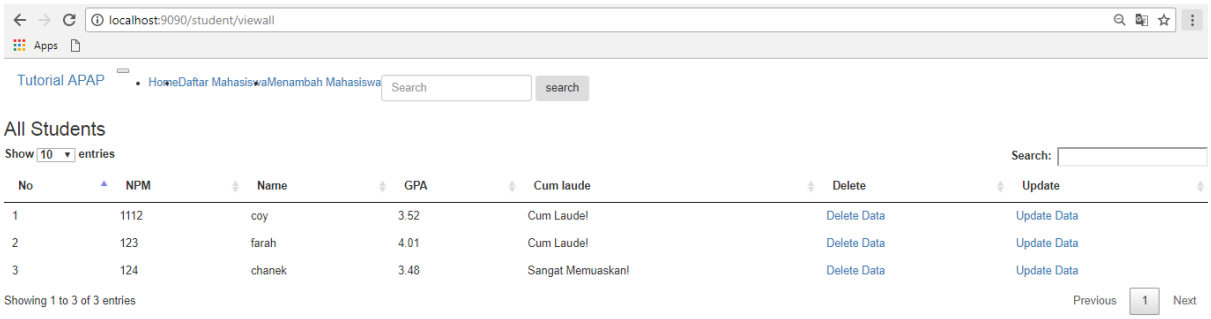
⇨ Sama fungsinya seperti object service yakni menghungkan dengan mapper dan model

StudentServiceRest

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

⇨ Sama fungsinya seperti object service database

Farah Agia Ramadhina
1506731580
APAP - A

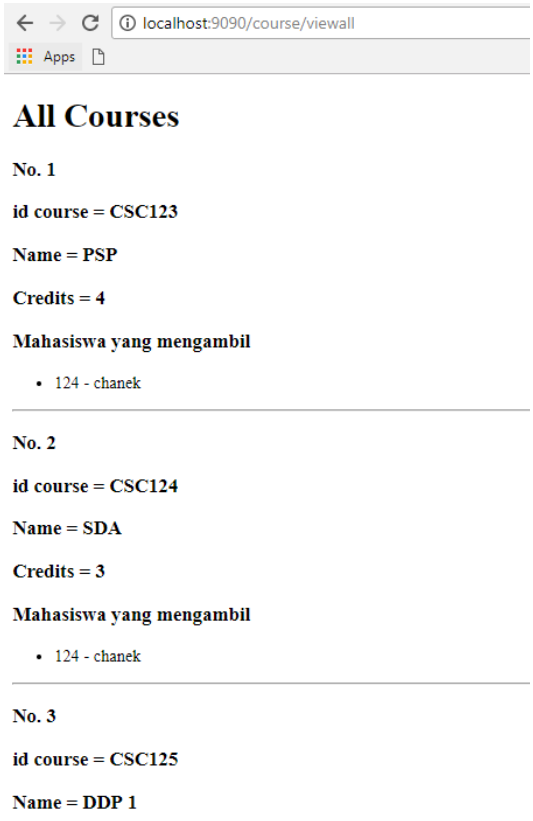


Penggunaan Rest adalah cara baru yang tidak perlu memparsing String melainkan object dan dengan cara ini juga bisa memetakan ke alamat URL sama halnya seperti Controller untuk kasus ini adalah melihat semua Student yang ada di database dengan mengantar ke halaman viewAll

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

Secara garis besar sama dengan yang di atas, maka berikut souce code & screenshot tertera

- **ViewAllCourses**



CourseDAOImpl

```
@Override
	public List<CourseModel> selectAllCourses() {
```

Farah Agia Ramadhina
1506731580
APAP - A

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

CourseDAO

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

CourseServiceRest

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

- **ViewCourse**

CourseDAOImpl

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

CourseDAO

```
public interface CourseDAO {
      CourseModel selectCourse(String idCourse);
}
```

CourseServiceRest

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

Summary: Pemisahan layer seperti ini memudahkan coder untuk melakukan coding tanpa perlu memikirkan database dengan adanya Producer (backend) dan Consumer (frontend) sehingga scope masing-masing layer batasannya lebih terlihat