

## 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"`

Dengan mencopy method `selectStudent` sebelumnya, kemudian method diubah dengan mengembalikan `arraylist` dari `students`.

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



```
localhost:8181/rest/student/viewall

[{"npm": "123", "name": "chanek", "gpa": 3.6, "courses": [{"idCourse": null, "name": "MPKT", "credits": 6, "students": null}]}, {"npm": "124", "name": "chanek Jr.", "gpa": 3.5, "courses": [{"idCourse": null, "name": "PSP", "credits": 4, "students": null}, {"idCourse": null, "name": "SDA", "credits": 3, "students": null}]}, {"npm": "1502132", "name": "Bagaskoro MMM", "gpa": 3.9, "courses": []}, {"npm": "151515", "name": "baba m", "gpa": 3.3, "courses": []}]
```

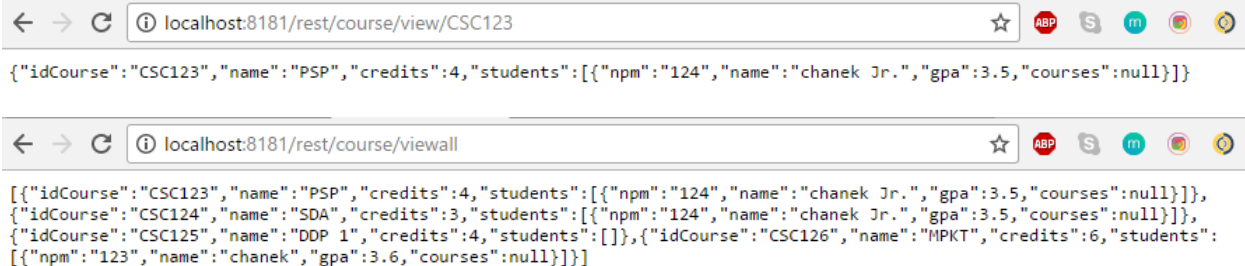
## Latihan 2:

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 elihat semua course (view all)

Method – method yang dibuat kurang lebih sama dengan method student untuk view dan viewall, hanya saja setiap “student” diubah menjadi course.

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

@RequestMapping("/course/viewall")
public List<CourseModel> viewallcourse () {
    List<CourseModel> students = studentService.selectAllCourses ();
    return students;
}
```



localhost:8181/rest/course/view/CSC123

```
{ "idCourse": "CSC123", "name": "PSP", "credits": 4, "students": [ { "npm": "124", "name": "chanek Jr.", "gpa": 3.5, "courses": null } ] }
```

localhost:8181/rest/course/viewall

```
[ { "idCourse": "CSC123", "name": "PSP", "credits": 4, "students": [ { "npm": "124", "name": "chanek Jr.", "gpa": 3.5, "courses": null } ] }, { "idCourse": "CSC124", "name": "SDA", "credits": 3, "students": [ { "npm": "124", "name": "chanek Jr.", "gpa": 3.5, "courses": null } ] }, { "idCourse": "CSC125", "name": "DDP 1", "credits": 4, "students": [ ] }, { "idCourse": "CSC126", "name": "MPKT", "credits": 6, "students": [ { "npm": "123", "name": "chanek", "gpa": 3.6, "courses": null } ] } ]
```

```
{
  "idCourse": "CSC123",
  "name": "PSP",
  "credits": 4,
  "students": [
    {
      "npm": "124",
      "name": "chanek Jr.",
      "gpa": 3.5,
      "courses": null
    }
  ]
}
```

```
[
  {
    "idCourse": "CSC123",
    "name": "PSP",
    "credits": 4,
    "students": [
      {
        "npm": "124",
        "name": "chanek Jr.",
        "gpa": 3.5,
        "courses": null
      }
    ]
  },
  {
    "idCourse": "CSC124",
    "name": "SDA",
    "credits": 3,
    "students": [
      {
        "npm": "124",
        "name": "chanek Jr.",
        "gpa": 3.5,
        "courses": null
      }
    ]
  },
  {
    "idCourse": "CSC125",
    "name": "DDP 1",
    "credits": 4,
    "students": []
  },
  {
    "idCourse": "CSC126",
    "name": "MPKT",
    "credits": 6,
    "students": [
      {
        "npm": "123",
        "name": "chanek",
        "gpa": 3.6,
        "courses": null
      }
    ]
  }
]
```

## Latihan 3:

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

Method dibuat pertama-tama pada section berikut:

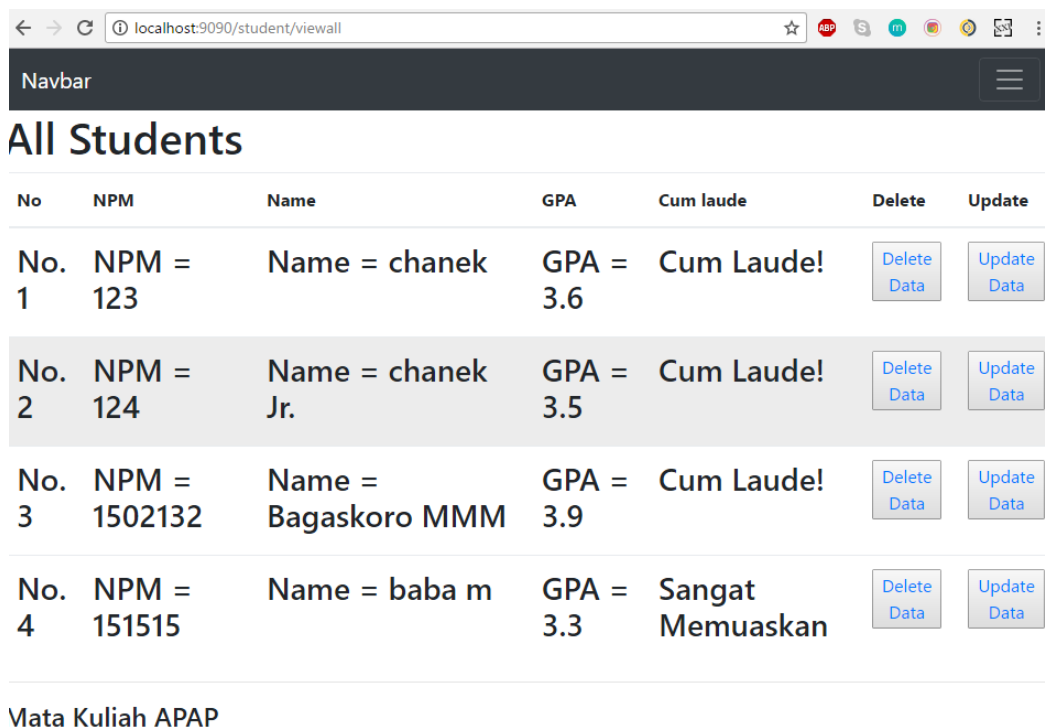
`StudentServiceRest`:

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

`StudentDAOImpl`:

```
@Override
public List<StudentModel> selectAllStudents ()
{
    ResponseEntity<List<StudentModel>> student =
        restTemplate.exchange(
            "http://localhost:8181/rest/student/viewall", HttpMethod.GET, null, new ParameterizedTypeReference<List<
        >>());
    List<StudentModel> students = student.getBody();
    return students;
}
```

Consumer view:



No	NPM	Name	GPA	Cum laude	Delete	Update
No. 1	NPM = 123	Name = chanek	GPA = 3.6	Cum Laude!	Delete Data	Update Data
No. 2	NPM = 124	Name = chanek Jr.	GPA = 3.5	Cum Laude!	Delete Data	Update Data
No. 3	NPM = 1502132	Name = Bagaskoro MMM	GPA = 3.9	Cum Laude!	Delete Data	Update Data
No. 4	NPM = 151515	Name = baba m	GPA = 3.3	Sangat Memuaskan	Delete Data	Update Data

Mata Kuliah APAP

## Latihan 4:

---

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

Method kurang lebih sama dengan Student untuk ServiceRest, DAO, dan DAOImpl nya. Kemudian membuat course/viewall pada course ccontroller untuk mengambil view yang dituju

CourseDAOImpl:

```
@Service
public class CourseDAOImpl implements CourseDAO
{
    @Bean
    public RestTemplate restTemplate(RestTemplateBuilder builder) {
        // Do any additional configuration here
        return builder.build();
    }

    @Autowired
    private RestTemplate restTemplate;

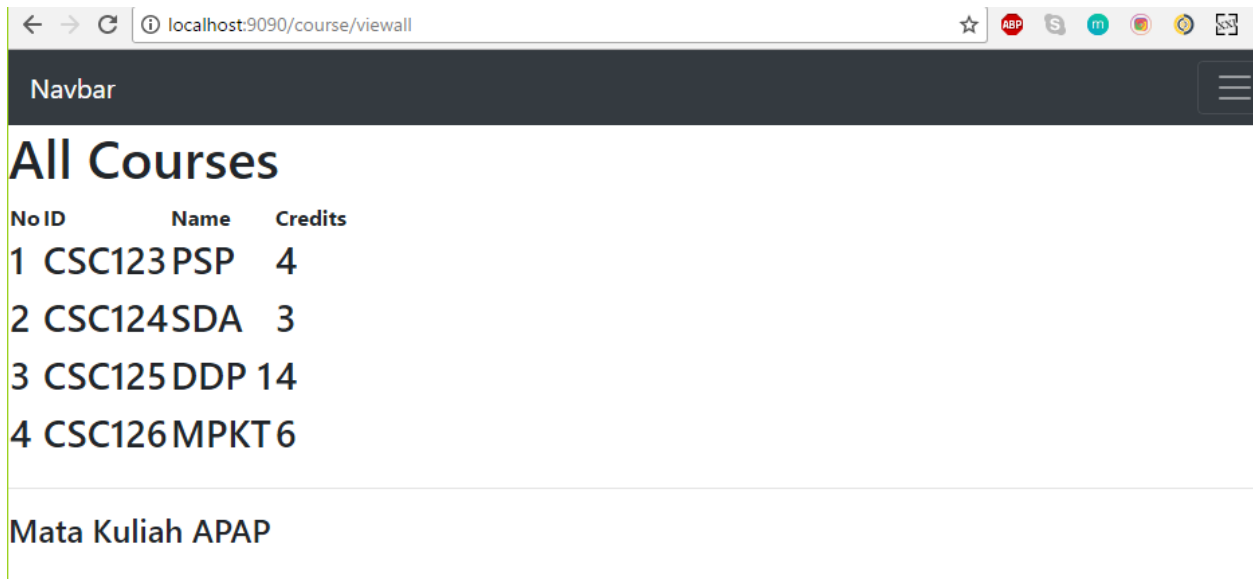
    @Override
    public CourseModel selectCourse (String id)
    {
        CourseModel course =
            restTemplate.getForObject(
                "http://localhost:8181/rest/course/view/"+id,
                CourseModel.class);
        System.out.println("Course " + course);
        return course;
    }

    @Override
    public List<CourseModel> selectAllCourses ()
    {
        ResponseEntity<List<CourseModel>> course =
            restTemplate.exchange(
                "http://localhost:8181/rest/course/viewall", HttpMethod.GET, null, new ParameterizedTypeReference<List<CourseModel>>() {
                });
        List<CourseModel> courses = course.getBody();
        System.out.println("Course " + courses);
        return courses;
    }
}
```

viewallCourse:

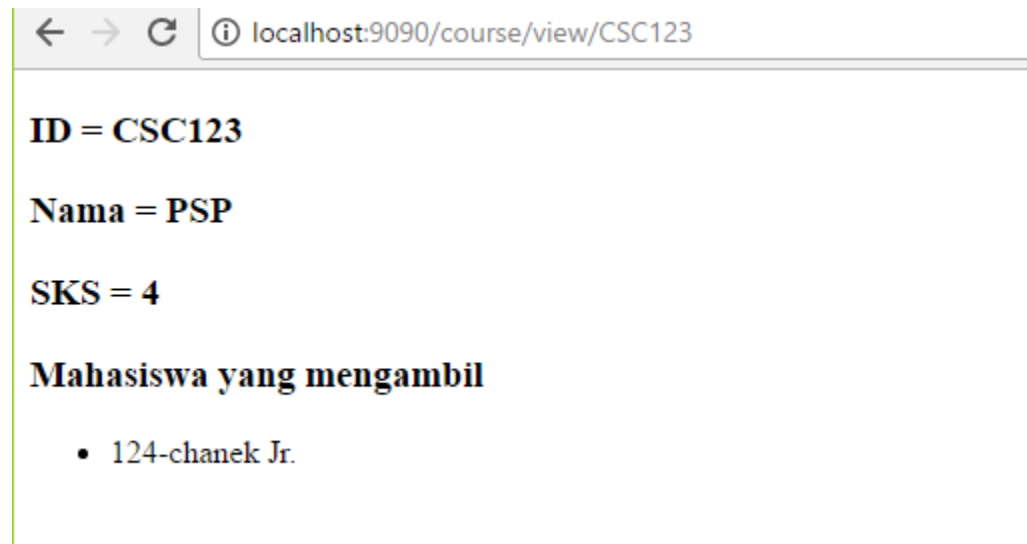
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <div th:replace="fragments/fragment :: links"></div>
  </head>
  <body>
    <div th:replace="fragments/fragment :: header"></div>
    <h1>All Courses</h1>
    <table id="table_id" class="display">
      <thead>
        <tr>
          <th>No</th>
          <th>ID</th>
          <th>Name</th>
          <th>Credits</th>
        </tr>
      </thead>
      <tbody>
        <div th:each="course, iterationStatus: ${courses}">
          <tr>
            <td><h3 th:text="${iterationStatus.count}">No. 1</h3></td>
            <td><h3 th:text="${course.idCourse}">Student NPM</h3></td>
            <td><h3 th:text="${course.name}">Student Name</h3></td>
            <td><h3 th:text="${course.credits}">Student GPA</h3></td>
          </tr>
        </div>
      </tbody>
    </table>
    <div th:replace="fragments/fragment :: footer"></div>
  </body>
</html>
```

View on browser



No ID	Name	Credits
1	CSC123 PSP	4
2	CSC124 SDA	3
3	CSC125 DDP	14
4	CSC126 MPKT	6

Mata Kuliah APAP



## NOTE

Pada screenshot saya menggunakan port 8181 untuk producer dikarenakan port 8080 pada local saya masih digunakan aplikasi lain. Untuk keperluan lab, commit pada git telah saya ubah ke port 8080 seperti yang tertulis pada instruksi. Terimakasih

## Pelajaran dari tutorial 7

---

Saya mempelajari bagaimana membuat webservice menggunakan Springboot. Webservice pada springboot menggunakan bantuan dari library RestTemplate untuk membuat REST API. Untuk menghubungkannya diperlukan REST yang diproduksi oleh webservice domain (dalam hal ini disebut producer) yang dibuat dari package .rest . Sedangkan bagi website yang ingin menggunakan web wervice tersebut harus men spesifikkan url REST pada DAO implementation di service.