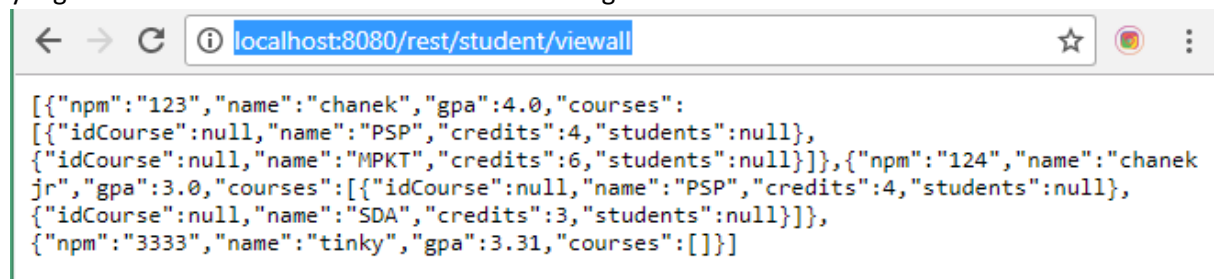


1. Service untuk mengembalikan seluruh *student* yang ada di basis data

Untuk mengembalikan seluruh *student* yang ada di basis data, saya membuat *method* `viewAll()` yang mengembalikan *list of student* sebagai berikut :

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

Method tersebut memanfaatkan *method* `selectAllStudents()` pada *class* `StudentService` yang sudah dibuat sebelumnya untuk melakukan *read* seluruh *student* pada basis data. Tampilan yang dihasilkan dari *method* tersebut adalah sebagai berikut:



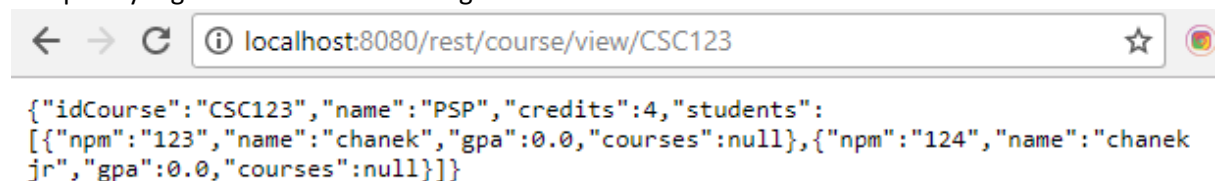
2. Service untuk *class* `Course`

- `view(id)`

Dalam *method* `view(id)`, saya mengimplementasikannya hampir mirip dengan `view(npm)` pada *class* `Student` sebelumnya. *Course* yang ingin ditampilkan akan di-*retrieve* menggunakan *method* `selectCourse(id)` yang telah diimplementasikan sebelumnya pada *class* `CourseService`. Berikut potongan kodenya:

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

Tampilan yang dihasilkan adalah sebagai berikut:



- `viewAll()`

Sebelumnya pada tutorial 5, saya belum mengimplementasikan *service* untuk me-*retrieve* seluruh *course* dari *database*. Karena itu, pertama-tama yang saya lakukan adalah membuat akses ke *database* untuk membaca seluruh *course* yang ada di *database*.

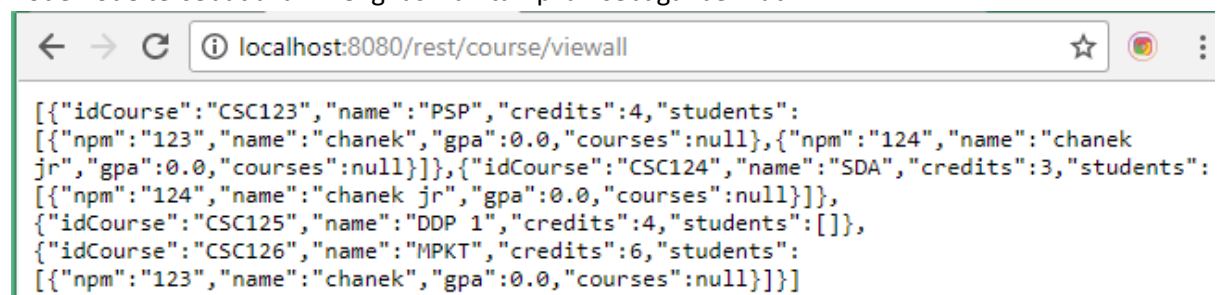
```
@Select("select id_course, name, credits from course")
@Results(value = {
    @Result(property="idCourse", column="id_course"),
    @Result(property="name", column="name"),
    @Result(property="credits", column="credits"),
    @Result(property="students", column="id_course",
        javaType = List.class,
```

```
        many=@Many(select="selectStudents")
    })
    List<CourseModel> selectAllCourses();
```

Setelah itu saya membuat *method* pengakses di *class* *CourseService* dan *CourseServiceDatabase*. Kemudian, pada *CourseRestController* saya membuat *method* *viewAll()* yang memanfaatkan *method* *selectAllCourses* pada *CourseService* yang baru saya buat tersebut. Berikut adalah *method* pada *class* *CourseRestService*:

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

Kode-kode tersebut akan menghasilkan tampilan sebagai berikut:



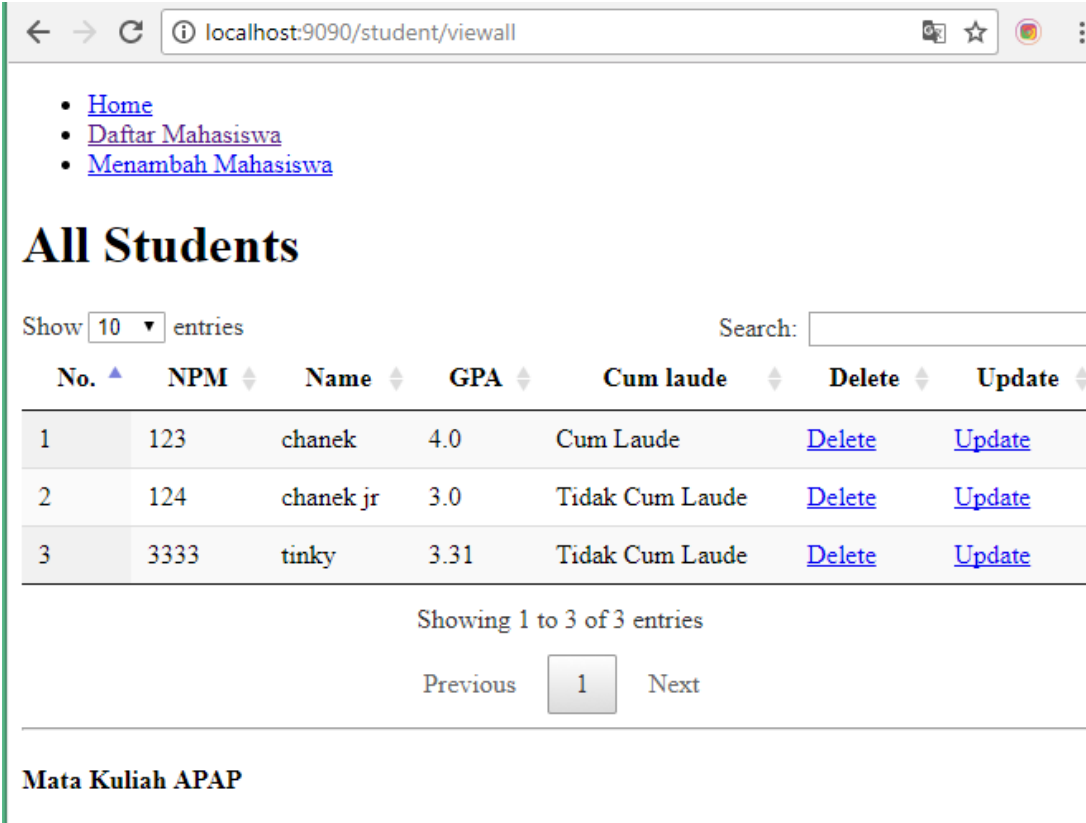
3. Implementasi *service consumer* untuk *view all* Students

Untuk memperoleh *viewAll* Students diperlukan *entity* untuk menampung *list of* *StudentModel*. Saya menggunakan tipe data *ResponseEntity<T>* untuk melakukan ini. *Generic type* *T* diisi dengan *List<StudentModel>*. Setelah menerima data dari *webservice* *Tutorial07Producer*, data akan ditampilkan. Potongan kode untuk melakukan *request* ke *webservice* adalah sebagai berikut:

```
@Override
public List<StudentModel> selectAllStudents() {
    ResponseEntity<List<StudentModel>> responseEntity =

        restTemplate.exchange("http://localhost:8080/rest/s
tudent/viewall",
            HttpMethod.GET,
            null,
            new ParameterizedTypeReference<List<StudentModel>>
                () {});
    List<StudentModel> students = responseEntity.getBody();
    return students;
}
```

Tampilan halaman akan menjadi sebagai berikut:



The screenshot shows a web browser at the URL `localhost:9090/student/viewall`. The page has a navigation menu with links: [Home](#), [Daftar Mahasiswa](#), and [Menambah Mahasiswa](#). Below the menu is a heading

All Students

. There is a search bar and a dropdown menu set to '10 entries'. A table displays student data with columns: No., NPM, Name, GPA, Cum laude, Delete, and Update. The table contains three rows of data. Below the table, it says 'Showing 1 to 3 of 3 entries' and has pagination links for 'Previous', '1', and 'Next'. At the bottom, there is a section titled 'Mata Kuliah APAP'.

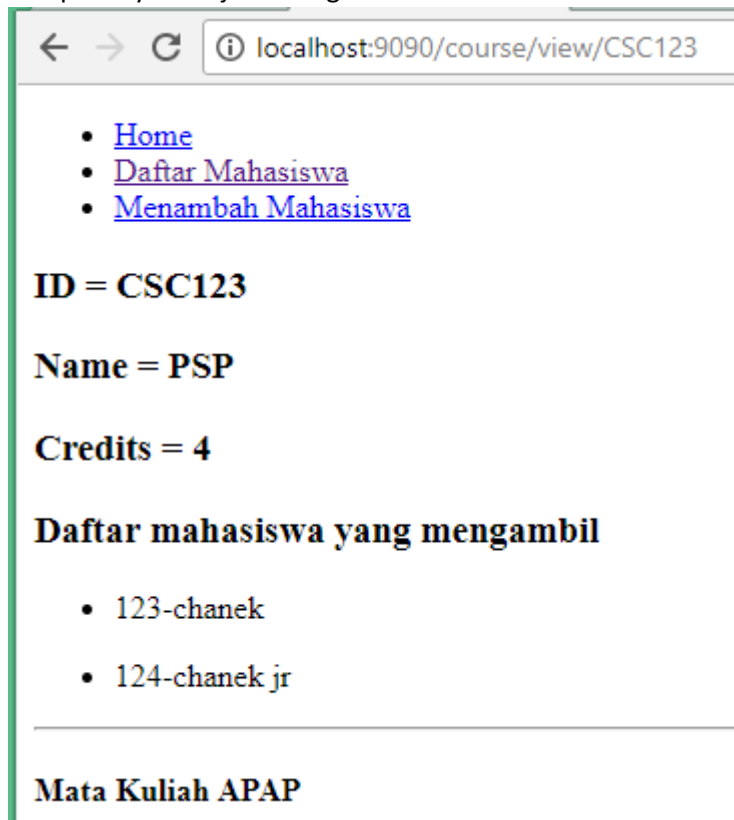
No.	NPM	Name	GPA	Cum laude	Delete	Update
1	123	chanek	4.0	Cum Laude	Delete	Update
2	124	chanek jr	3.0	Tidak Cum Laude	Delete	Update
3	3333	tinky	3.31	Tidak Cum Laude	Delete	Update

4. Implementasi *service consumer* untuk *class CourseModel*

- view(id)
Method ini kurang lebih sama dengan view(npm) yang telah dijelaskan pada tutorial, dimana *course* dengan id yang diminta akan ditampilkan. Pertama-tama, saya membuat *interface CourseDAO* yang berisi *method* *selectCourse(id)* dan *selectAllCourses()*. Kemudian *interface* tersebut diimplementasikan dalam *class StudentDAOImpl.java*. Implementasinya *method selectCourse(id)* adalah sebagai berikut:

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

Course di-retrieve melalui *method* `getForObject()`. *Method* tersebut dimiliki oleh *class* `RestTemplate` dari `Tutorial07Producer` yang akan menampilkan *course* yang dimaksud. Tampilannya menjadi sebagai berikut:



- `viewAll()`
Untuk menampilkan seluruh *courses* yang ada, saya menggunakan *method* `exchange()` milik `RestTemplate` untuk me-*retrieve* data dari `Tutorial07Producer`.

```
@Override
public List<CourseModel> selectAllCourses() {
    ResponseEntity<List<CourseModel>> responseEntity =
        restTemplate.exchange("http://localhost:8080/rest/c
        ourse/viewall",
            HttpMethod.GET,
            null,
            new ParameterizedTypeReference<List<CourseModel>>()
            {});
    List<CourseModel> courses = responseEntity.getBody();
    return courses;
}
```

Tampilan menjadi sebagai berikut:

- [Home](#)
- [Daftar Mahasiswa](#)
- [Menambah Mahasiswa](#)

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

1

Next

Mata Kuliah APAP