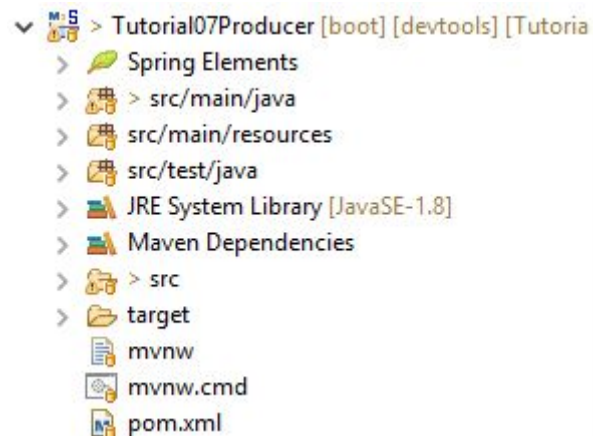


Tutorial 7

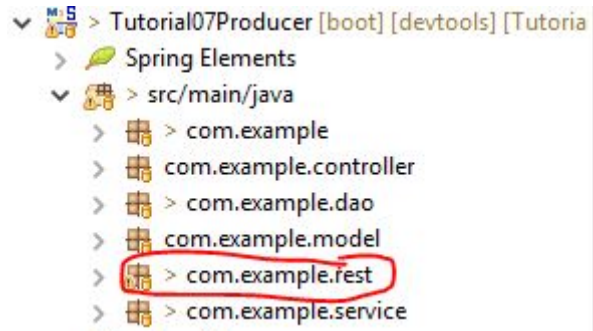
Membuat Web Service Menggunakan Spring Boot Framework

Membuat Service Producer

1. Import berkas-berkas yang ada di tutorial 5 untuk Service Producer ini. Buatlah project 'Tutorial07Producer' dan silakan import satu folder atau import file satu per satu dari Tutorial05.



2. Buat *package* baru `com.example.rest`, atau sesuaikan dengan *package* Anda masing-masing REST Controller Web Service diletakkan pada *package* terpisah, bukan pada *package* `com.example.controller`



3. Buat *class* baru yaitu `StudentRestController.java` pada *package* `com.example.rest`

```
1 package com.example.rest;
2
3 import java.util.List;
4
5 @RestController
6 @RequestMapping("/rest")
7 public class StudentRestController {
8
9     @Autowired
10     StudentService studentService;
11
12     @RequestMapping("/student/view/{npm}")
13     public StudentModel view (@PathVariable(value = "npm") String npm) {
14         StudentModel student = studentService.selectStudent (npm);
15         return student;
16     }
17 }
```

Tampilan ketika membuka “localhost:8080/rest/student/view/124”



```
1 // 20171104190815
2 // http://localhost:8080/rest/student/view/124
3
4 {
5   "npm": "124",
6   "name": "Gheafany Widyatika Putri",
7   "gpa": 3.4,
8   "courses": [
9     {
10       "id_course": "CSC123",
11       "name": "PSP",
12       "credits": 4.0,
13       "students": null
14     },
15     {
16       "id_course": "CSC124",
17       "name": "SDA",
18       "credits": 3.0,
19       "students": null
20     }
21   ]
22 }
```

LATIHAN

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”.

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

Penjelasan Method :

Method diatas akan dieksekusi ketika *end point* “/rest/student/viewall” diakses melalui URL, ketika diakses maka producer akan mengembalikan data *students* di dalam *database* dalam bentuk JSON yang nantinya akan di-consume oleh *consumer*.

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

Penjelasan Method:

Method diatas ketika dipanggil akan mengeksekusi *query* di kelas *mapper* yang akan mengembalikan semua data *students* di dalam *database* dalam bentuk *List of Students*.

Tampilan keluarannya:



```
13 {
14   "npm": "124",
15   "name": "Gheafany Widyatika Putri",
16   "gpa": 3.4,
17   "courses": [
18     {
19       "id_course": "CSC123",
20       "name": "PSP",
21       "credits": 4.0,
22       "students": null
23     },
24     {
25       "id_course": "CSC124",
26       "name": "SDA",
27       "credits": 3.0,
28       "students": null
29     }
30   ]
31 },
32 {
33   "npm": "1506689515",
34   "name": "Jihan Adibah Ba'abud",
35   "gpa": 3.85,
36   "courses": [
37
38   ]
39 },
40 {
41   "npm": "1506689686",
42   "name": "Aisyah Husna",
43   "gpa": 3.5,
44   "courses": [
45
46   ]
47 },
```

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*).

View by ID

```
@Override
public CourseModel selectCourse(String id_course) {
    return studentMapper.selectCourse(id_course);
}
```

Penjelasan Method:

Method tersebut akan mengeksekusi *query* yang akan mengembalikan *detail* data suatu *course* pada *database* dalam bentuk objek CourseModel

```
1 package com.example.rest;
2
3 import java.util.List;
14
15 @RestController
16 @RequestMapping("/rest")
17 public class CourseRestController {
18
19
20     @Autowired
21     StudentService studentService;
22
23     @RequestMapping("/course/view/{npm}")
24     public CourseModel view (@PathVariable(value = "npm") String npm) {
25         CourseModel course = studentService.selectCourse (npm);
26
27         return course;
28     }
}
```

Penjelasan Method:

Method tersebut akan diakses ketika *end point* `/rest/course/view/{npm}` diakses melalui URL. Ketika diakses *method* tersebut akan mengembalikan *detail* data suatu Course dalam bentuk JSON

```
@RequestMapping("/course/viewall")
public List<CourseModel> view (Model model)
{
    List<CourseModel> courses = studentService.selectAllCourses ();
    return courses;
}
```

Penjelasan Method:

Method tersebut akan dieksekusi ketika *end point* `/rest/course/viewall` diakses melalui URL. *Method* tersebut akan mengembalikan semua *data course* di dalam *database* dalam bentuk JSON.

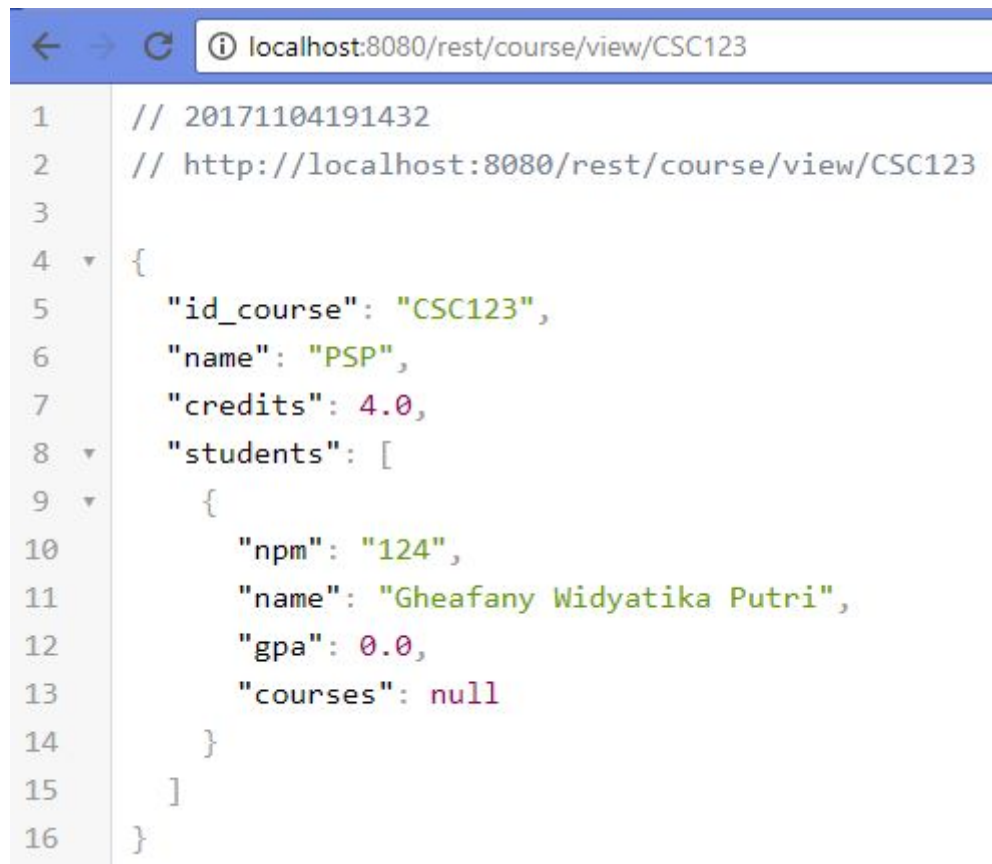
View All

```
@Override  
public List<CourseModel> selectAllCourses() {  
    return studentMapper.selectAllCourses();  
}
```

Penjelasan Method:

Method tersebut akan mengeksekusi *query* pada *class mapper* dan akan mengembalikan semua *data course* yang terdapat didalam *database* dalam bentuk *list of CourseModel*

Tampilan *output* ketika *endpoint* /rest/course/view/CSC123 diakses



```
localhost:8080/rest/course/view/CSC123  
  
1 // 20171104191432  
2 // http://localhost:8080/rest/course/view/CSC123  
3  
4 {  
5   "id_course": "CSC123",  
6   "name": "PSP",  
7   "credits": 4.0,  
8   "students": [  
9     {  
10      "npm": "124",  
11      "name": "Gheafany Widyatika Putri",  
12      "gpa": 0.0,  
13      "courses": null  
14    }  
15  ]  
16 }
```

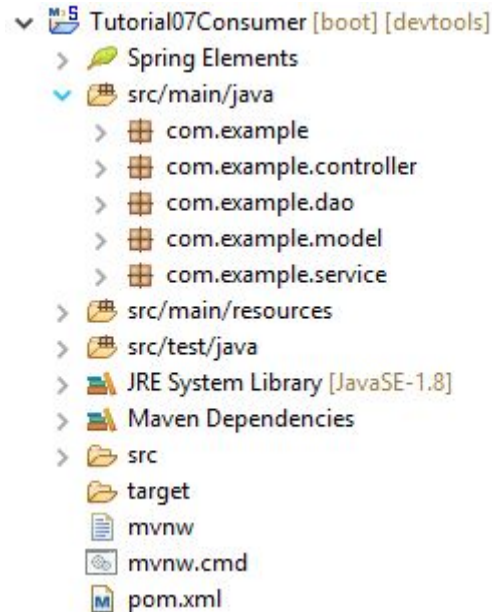
Tampilan *output* ketika *end point* /rest/course/viewall diakses



```
1 // 20171104191615
2 // http://localhost:8080/rest/course/viewall
3
4 [
5   {
6     "id_course": "CSC123",
7     "name": "PSP",
8     "credits": 4.0,
9     "students": [
10      {
11        "npm": "124",
12        "name": "Gheafany Widyatika Putri",
13        "gpa": 0.0,
14        "courses": null
15      }
16    ]
17  },
18  {
19    "id_course": "CSC124",
20    "name": "SDA",
21    "credits": 3.0,
22    "students": [
23      {
24        "npm": "124",
25        "name": "Gheafany Widyatika Putri",
26        "gpa": 0.0,
27        "courses": null
28      }
29    ]
30  },
31  {
32    "id_course": "CSC125",
33    "name": "DDP 1",
34    "credits": 4.0,
35    "students": [
36
37    ]
38  },
39  {
40    "id_course": "CSC126",
41    "name": "MPKT".
```


Membuat Service Consumer

1. Import berkas-berkas yang ada di tutorial 6 untuk *Service Consumer* ini. Buatlah project 'Tutorial07Consumer' dan silakan *import* satu folder atau *import file* satu per satu dari Tutorial06.



2. Karena *Service Producer* dan *Service Consumer* harus dijalankan secara bersamaan, ubah berkas *application.properties* yang ada di folder *resources* dengan menambahkan baris *server.port=9090*. Nantinya Aplikasi *Service Producer* akan dijalankan di *localhost:8080* sedangkan *Service Consumer* akan dijalankan di *localhost:9090*.

```
1 # Data Source
2 spring.datasource.platform=mysql
3 spring.datasource.driver-class-name=com.mysql.jdbc.Driver
4 spring.datasource.url=jdbc:mysql://localhost:3306/eaap
5 spring.datasource.username=root
6 spring.datasource.password=
7 spring.datasource.initialize=false
8
9 server.port:9090
```


3. Di DAO tambahkan interface dengan nama StudentDAO

```
1 package com.example.dao;  
2  
3 import java.util.List;  
4  
5  
6  
7 public interface StudentDAO {  
8  
9     StudentModel selectStudent (String npm);  
10    List<StudentModel> selectAllStudents ();  
11  
12 }
```

4. Selanjutnya kita akan membuat implementasi kelas StudentDAO tersebut dengan nama kelas StudentDAOImpl.java. Isinya adalah sebagai berikut

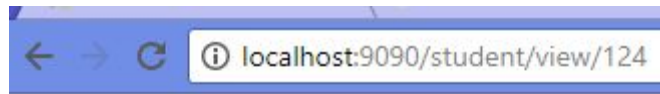
```
14 @Service  
15 public class StudentDAOImpl implements StudentDAO {  
16  
17     @Autowired  
18     private RestTemplate restTemplate;  
19  
20     @Override  
21     public StudentModel selectStudent (String npm)  
22     {  
23         StudentModel student =  
24             restTemplate.getForObject(  
25                 "http://localhost:8080/rest/student/view/"+npm,  
26                 StudentModel.class);  
27         return student;  
28     }  
29  
30     @Override  
31     public List<StudentModel> selectAllStudents ()  
32     {  
33         ResponseEntity<List<StudentModel>> rateResponse =  
34             restTemplate.exchange("http://localhost:8080/rest/student/viewall",  
35                 HttpMethod.GET, null, new ParameterizedTypeReference<List<StudentModel>>() {  
36  
37             });  
38         List<StudentModel> students = rateResponse.getBody();  
39         return students;  
40     }  
41  
42     @Bean  
43     public RestTemplate restTemplate() {  
44         return new RestTemplate();  
45     }  
46 }
```

5. Selanjutnya, kita ingin mengubah agar `StudentService` mengambil data dari *web service* bukan dari *database*. Kita tidak perlu menghapus *class* `StudentServiceDatabase`. Karena *scalable system*, kita cukup menambahkan *class* baru yaitu `StudentServiceRest` yang *implement* `StudentService` di *package service*. Isinya adalah sebagai berikut:



```
StudentDAO.java StudentDAOIm... *StudentServ... CourseDAO
16 @Slf4j
17 @Service
18 @Primary
19 public class StudentServiceRest implements StudentService {
20
21     @Autowired
22     private StudentDAO studentDAO;
23
24     @Autowired
25     private CourseDAO courseDAO;
26
27     @Override
28     public StudentModel selectStudent(String npm) {
29         log.info ("REST - select student with npm {}", npm);
30         return studentDAO.selectStudent (npm);
31     }
32
33     @Override
34     public List<StudentModel> selectAllStudents() {
35         log.info ("REST - select all students");
36         return studentDAO.selectAllStudents();
37     }
38
39     @Override
40     public void addStudent(StudentModel student) {}
41
42     @Override
43     public void deleteStudent(String npm) {}
44
45     @Override
46     public void updateStudent(StudentModel student) {}
47
48     @Override
```

6. Tampilan setelah membuka “localhost:9090/student/view/124”



NPM = 124

Name = Gheafany Widyatika Putri

GPA = 3.4

Kuliah yang diambil

- PSP-4.0 sks
- SDA-3.0 sks

LATIHAN

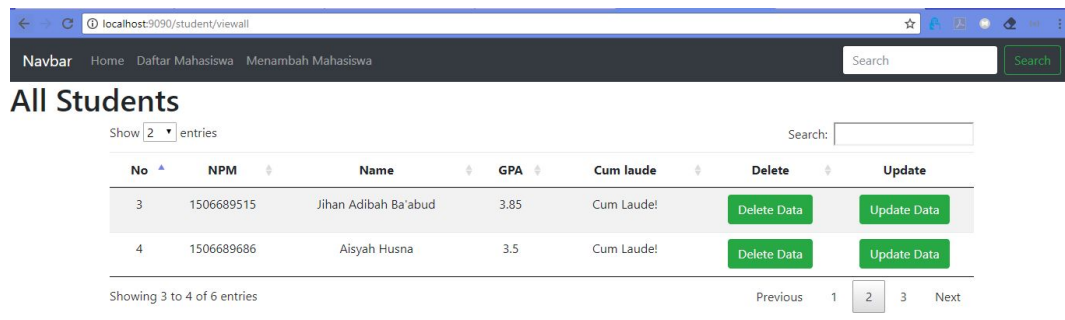
Latihan 3: Implementasikan *service consumer* untuk *view all Students* dengan melengkapi *method* `selectAllStudents` yang ada di kelas `StudentServiceRest`.

```
@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;
}

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

Penjelasan *Method*:

Method tersebut berfungsi untuk mengakses *web service* dengan *end point* `/rest/student/viewall` yang dimana akan menerima data semua *student* di dalam *database* dalam bentuk JSON yang di-convert menjadi *list of students*



No	NPM	Name	GPA	Cum laude	Delete	Update
3	1506689515	Jihan Adibah Ba'abud	3.85	Cum Laude!	Delete Data	Update Data
4	1506689686	Aisyah Husna	3.5	Cum Laude!	Delete Data	Update Data

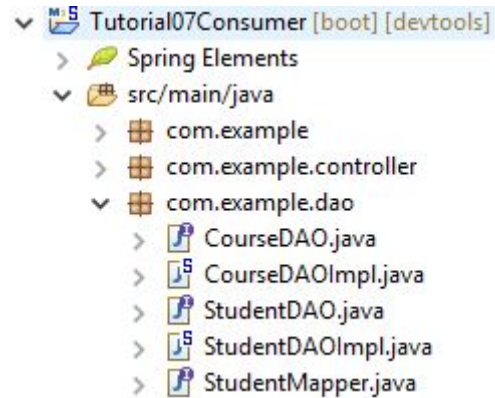
Mata Kuliah APAP

Data semua *students* dipanggil melalui `StudentServiceRest`

```
INFO 11800 --- [nio-9090-exec-4] com.example.service.StudentServiceRest : REST - select student with npm 11121313
INFO 11800 --- [nio-9090-exec-6] com.example.service.StudentServiceRest : REST - select all students
INFO 11800 --- [nio-9090-exec-8] com.example.service.StudentServiceRest : REST - select student with npm 11121313
INFO 11800 --- [nio-9090-exec-9] com.example.service.StudentServiceRest : REST - select all students
INFO 11800 --- [nio-9090-exec-1] com.example.service.StudentServiceRest : REST - select student with npm 11121313
INFO 11800 --- [nio-9090-exec-5] com.example.service.StudentServiceRest : REST - select all students
INFO 11800 --- [nio-9090-exec-6] com.example.service.StudentServiceRest : REST - select all students
INFO 11800 --- [nio-9090-exec-9] com.example.service.StudentServiceRest : REST - select all students
```

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

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



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

Penjelasan Method:

Method tersebut berfungsi untuk menginisialisasi restTemplate secara otomatis menggunakan bantuan anotasi @Autowired

```
15 @Service
16 public class CourseDAOImpl implements CourseDAO {
17
18     @Autowired
19     private RestTemplate restTemplate;
20
21     @Override
22     public CourseModel selectCourse(String id_course) {
23         CourseModel course =
24             restTemplate.getForObject(
25                 "http://localhost:8080/rest/course/view/"+ id_course,
26                 CourseModel.class);
27         return course;
28     }
29 }
```

Penjelasan Method:

Method tersebut digunakan untuk mengambil detail data sebuah *course* dalam bentuk JSON dengan mengakses *end point* “/rest/course/view”.

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

Penjelasan *Method*:

Method tersebut digunakan untuk mengambil semua *data course* dari *database* dalam bentuk JSON dengan mengakses *end point* “/rest/course/viewall”.

Hal yang saya pelajari di tutorial ini

Pada tutorial ini saya belajar mengenai penggunaan *web service* pada *spring framework*, belajar bagaimana cara untuk mengirimkan data dari *database* ke dalam bentuk JSON yang nantinya akan disimpan pada suatu *end point* sehingga dapat diakses oleh *consumer*, dan belajar bagaimana dua buah *project spring* dapat berinteraksi satu sama lain.