

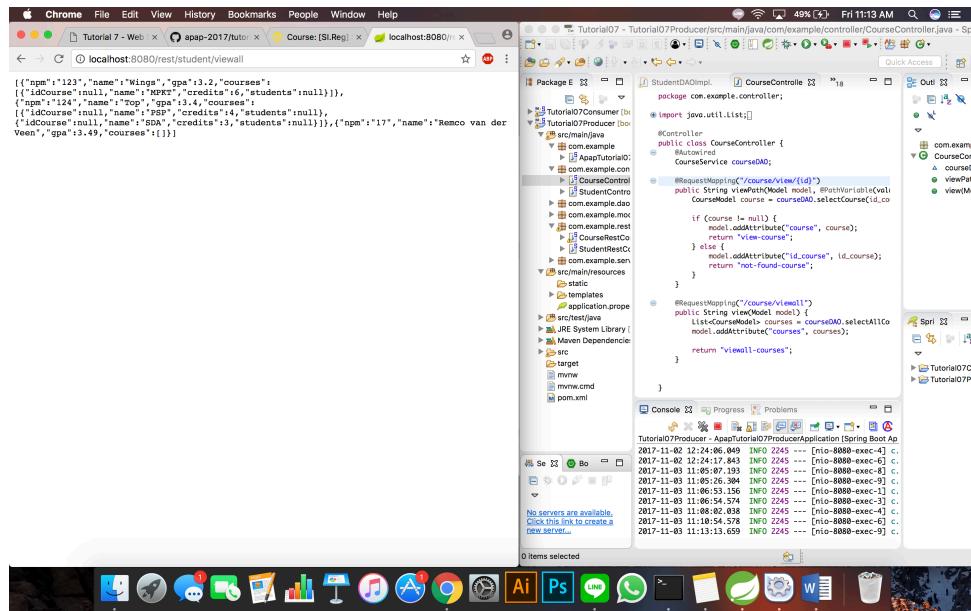
**Sum Up yang dipelajari pada Tutorial 7:** Hal yang saya pelajari pada Tutorial 7 adalah beberapa hal sebagai berikut – yang pertama yaitu pengimplementasian pemisahan *layer backend* (*service producer*) dan *layer frontend* (*service consumer*). Dengan adanya pemisahan tanggung jawab ini, sebuah aplikasi *service consumer* dapat fokus untuk menyediakan dan mengolah data saja ke pengguna tanpa perlu memiliki database. Sementara *service producer* hanya bertanggung jawab untuk menyediakan data dari *database* dan biasanya tidak memiliki view yang dapat dilihat pengguna.

**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 String view(Model model) {
    List<StudentModel> students = studentDAO.selectAllStudents();
    model.addAttribute("students", students);

    return "viewall";
}
```

Contoh keluaran <http://localhost:8080/rest/student/viewall>



**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).

```
@RequestMapping("/course/view/{id}")
public String viewPath(Model model, @PathVariable(value = "id") String id_course) {
    CourseModel course = courseDAO.selectCourse(id_course);

    if (course != null) {
        model.addAttribute("course", course);
```

```

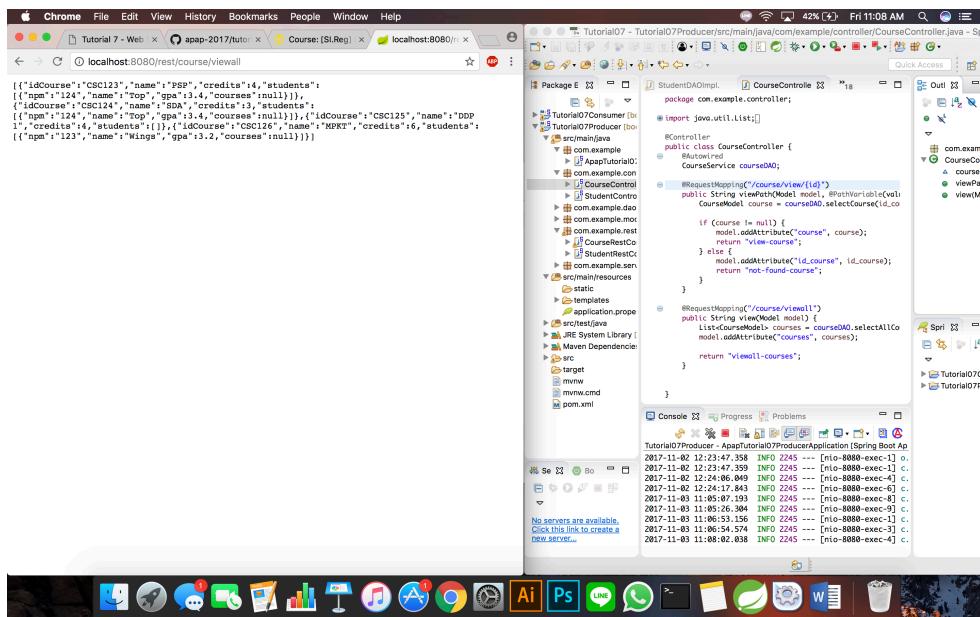
        return "view-course";
    } else {
        model.addAttribute("id_course", id_course);
        return "not-found-course";
    }
}

@RequestMapping("/course/viewall")
public String view(Model model) {
    List<CourseModel> courses = courseDAO.selectAllCourses();
    model.addAttribute("courses", courses);

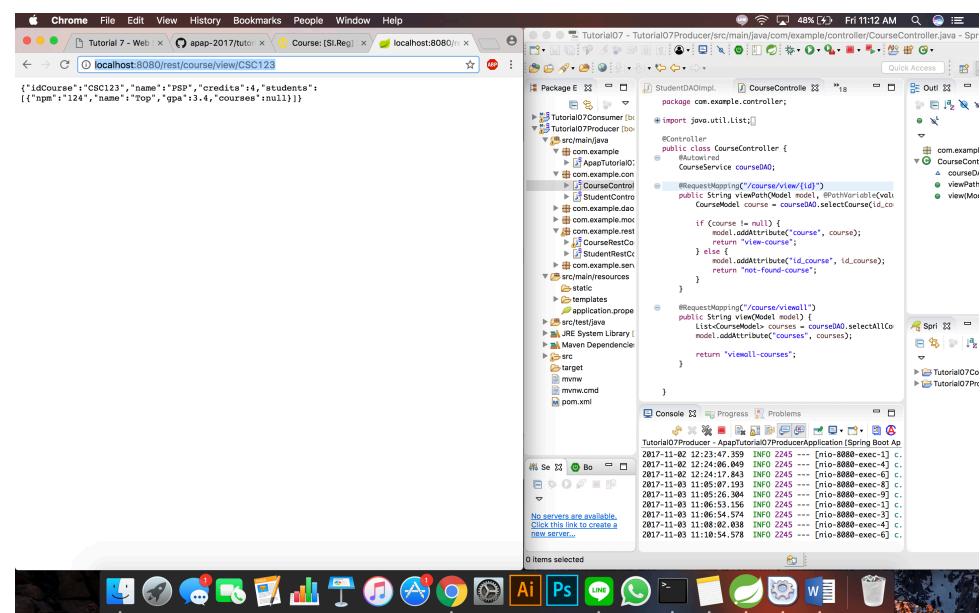
    return "viewall-courses";
}

```

Contoh keluaran <http://localhost:8080/rest/course/viewall>



Contoh keluaran <http://localhost:8080/rest/course/view/CSC123>



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

Pada file StudentDAOImpl.java:

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

Pada file StudentServiceRest.java:

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

Penambahan pada file ApapTutorial07ConsumerApplication.java:

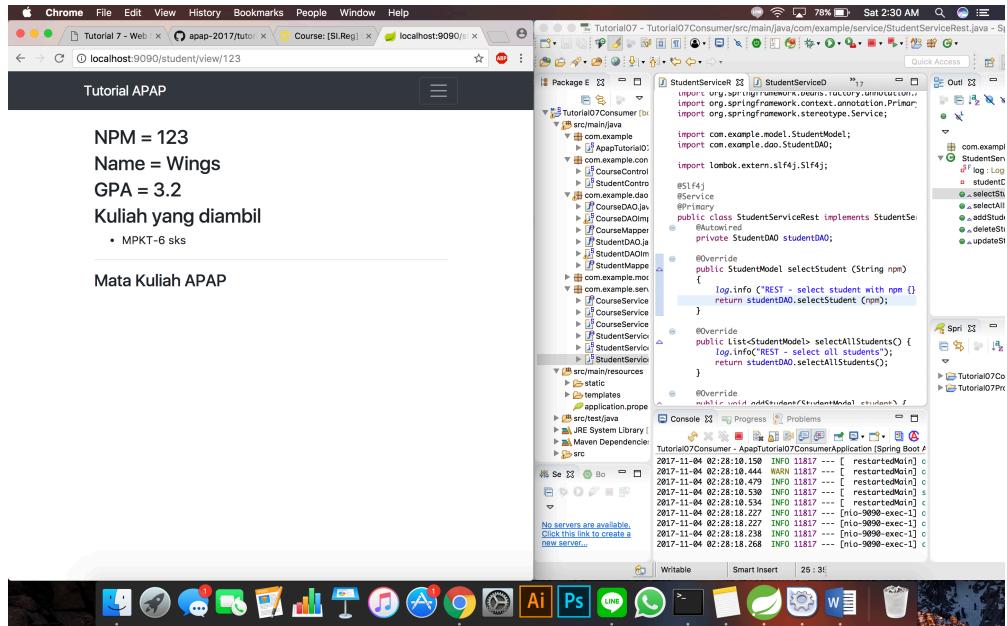
```
// create new bean for restTemplate  
// https://stackoverflow.com/questions/28024942/how-to-autowire-resttemplate-using-  
annotations  
@Bean  
public RestTemplate restTemplate() {  
    return new RestTemplate();  
}
```

Contoh keluaran <http://localhost:9090/student/viewall>

The screenshot shows a Mac OS X desktop with a browser window titled "Tutorial 7 - Web" displaying the URL "localhost:9090/student/viewall". The page shows a table titled "All Students" with three rows of student data. Below the table, it says "Showing 1 to 3 of 3 entries". The Java IDE (IntelliJ IDEA) has a package structure on the left and the code for "StudentServiceRest.java" on the right. The code implements the "StudentService" interface and overrides the "selectAllStudents" method to call the "selectAllStudents" method from the "studentDAO" bean. The code also includes logging statements using the "log.info" method.

```
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Primary;  
import org.springframework.stereotype.Service;  
  
import com.example.model.StudentModel;  
import com.example.dao.StudentDAO;  
  
import lombok.extern.slf4j.Slf4j;  
  
@Service  
@Primary  
public class StudentServiceRest implements StudentService {  
    @Autowired  
    private StudentDAO studentDAO;  
  
    @Override  
    public StudentModel selectStudent(String rpm) {  
        log.info("REST - select student with rpm {}");  
        return studentDAO.selectStudent(rpm);  
    }  
  
    @Override  
    public List<StudentModel> selectAllStudents() {  
        log.info("REST - select all students");  
        return studentDAO.selectAllStudents();  
    }  
  
    @Override  
    public void updateStudent(StudentModel student) {  
        log.info("REST - update student with studentId {}");  
        studentDAO.updateStudent(student);  
    }  
}
```

Contoh keluaran <http://localhost:9090/student/view/123>



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

Membuat file baru yaitu `CourseDAO.java`, dan `CourseDAOImpl.java`. Caranya sama dengan latihan no.3 namun untuk file course.

Pada file `CourseDAOImpl.java`:

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

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

Pada file CourseServiceRest.java:

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

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

Contoh keluaran <http://localhost:9090/course/view/CSC123>

