Ari Tri Wibowo Y

1506735175

APAP - C

Tutorial 7

Membuat Web Service Menggunakan Spring Boot Framework

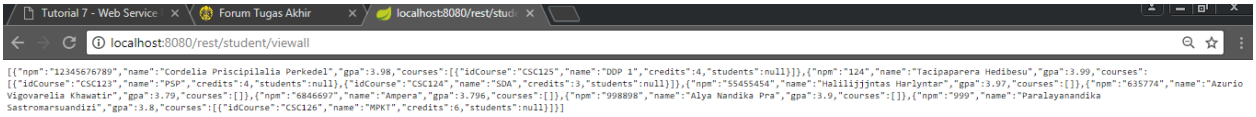
***

Hal yang dipelajari dalam tutorial ini

Hal yang dipelajari dalam tutorial 7 matakuliah APAP yakni bagaimana cara memisahkan layar backend (service producer) dan layer frontend (service consumer). Service producer akan memberikan data sementara itu service consumer akan meminta dan menerima data lalu mengolahnya tanpa perlu terhubung dengan database karena yang bertanggung jawab untuk urusan database adalah si service producer. Dalam tutorial ini, service producer akan memproduksi data dalam format JSON dan disajikan dalam sebuah URL yang nantinya akan diminta oleh service consumer.

Latihan 1

Service untuk mengembalikan seluruh student yang ada di database



Dimapping ke /rest/student/viewall

```
@RequestMapping("/student/viewall")
public List<StudentModel> viewall () {

    List<StudentModel> students = studentDAO.selectAllStudents();
    return students;
}
```

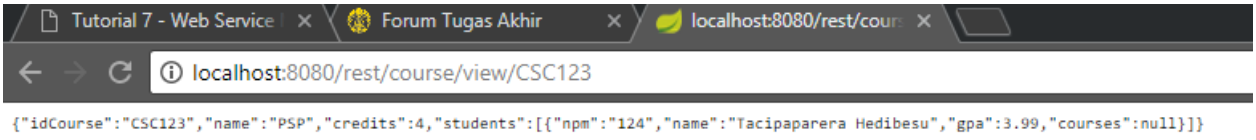
Penjelasan method

Dalam StudentRestController dibuat terlebih dahulu anotasi @Autowired StudentService studentDAO

Dengan autowired ini, maka bisa memanggil method selectAllStudents() yang ada pada objek studentDAO lalu disimpan dalam List berisikan StudentModel. List ini bernama students. Kemudian akan di-return nilai dari list ini.

Latihan 2

Service untuk melihat suatu course dengan masukan ID Course




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

Method ini akan menangani ketika diakses path /course/view/{idcourse} lalu akan menyimpan nama variable idcourse tersebut. Kemudan courseService akan mencari course di database dengan input parameter idcourse dan akan

disimpan dalam objeck dari kelas CourseModel. Akan direturn object course itu sendiri.

Service untuk melihat semua course


```java
@RequestMapping("/course/viewall")
public List<CourseModel> viewall () {

    List<CourseModel> courses = courseService.selectAllCourses();
    return courses;
}
```

Method ini akan menangani ketika diakses path /course/viewall, kemudian courseService akan memanggil method selectAllCourses() lalu disimpan dalam sebuah List yang berisikan objek-objek dengan tipe CourseModel. Method ini akan mengembalikan List yang bernama courses.

Latihan 3

Implementasi serviceconsumer untuk viewall Students pada kelas StudentDAOImpl.java

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

Dalam method ini, restTemplate akan mengambil objek pada link http://localhost:8080/rest/student/viewall yang dihasilkan dari service producer, kemudian akan disimpan dalam sebuah List yang berisikan objek-objek StudentModel. List ini bernama students. Method ini akan mereturn list students.

Implementasi serviceconsumer untuk viewall Students pada kelas StudentServiceRest.java

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

Method ini akan mereturn sebuah List yang berisikan ojek-objek dari StudentModel dengan menjalankan perintah selectAllStudents yang ada pada studentDAO.

Latihan 4

Implementasi service consumer untuk class CourseModel dengan membuat class-class DAO dan service baru

Membuat interface CourseDAO.java

```java
package com.example.dao;

import java.util.List;

import com.example.model.CourseModel;

public interface CourseDAO {

    CourseModel selectCourse(String idCourse);
    List<CourseModel> selectAllCourses();

}
```

Interface ini berisikan method yang harus ada pada kelas yang mengimplementasikan interface ini yakni selectCourse dan selectAllCourses

Membuat class CourseDAOImpl yang mengimplement interface CourseDAO

```java
public class CourseDAOImpl implements CourseDAO {

    @Autowired
    private RestTemplate restTemplate;

    @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;
    }

}
```

Class CourseDAOImpl memiliki instant variable restTemplate, method selectCourse, dan selectAllCourses. Method selectCourse akan menjalankan restTemplate dan mengambil objek pada url yang target lalu menyimpannya dalam sebuah objek CourseModel bernama course. Lalu akan mereturn course tersebut.

Membuat class CourseServiceRest

```
public class CourseServiceRest implements CourseService{

    @Autowired
    private CourseDAO courseDAO;

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

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


}
```

Class CourseServiceRest mengimplementasikan interface CourseService. Class ini memiliki instant variable CourseDAO. Method di dalamnya yakni method selectCourse yang akan menerima masukan berupa idCourse.