Sandar Klinton - 1306404720
APAP-A, Tutorial 7

Yang didapatkan pada Tutorial 7:

1. Mengetahui tentang pembuatan REST Framework pada Spring Boot
2. Mengetahui bahwa REST juga bisa digunakan untuk mengantarkan objek
3. RestTemplate pada Spring bisa digunakan untuk POST

Latihan

1. 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".
   -Pada StudentRestController:

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

   List akan dikembalikan oleh Spring dengan format JSON.
   Beberapa edit dilakukan pada Mapper untuk mengassign nilai id_course kepada field idCourse pada class StudentModel.
2. 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).

```
package com.example.rest;

import java.util.List;

@RestController
@RequestMapping("/rest")
public class CourseRestController{

    @Autowired
    StudentService studentService;

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

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

   Membuat sebuah Controller untuk Course. Disini, semua mapping akan langsung mengembalikan objek dengan bentuk JSON.

```
@Select("select * from course where id_course = #{id}")
@Results(value= {
        @Result(property="idCourse", column="id_course"),
        @Result(property="name", column="name"),
        @Result(property="credit", column="credit"),
        @Result(property="students", column="id_course",javaType = List.class, many=@Many(select="selectStudentsCourse"))
})
CourseModel selectCourse (@Param("id") String id);

@Select("select id_course, name, credits from course")
@Results(value= {
        @Result(property="idCourse", column="id_course")
})
List<CourseModel> selectAllCourse();
```

Diatas adalah Mapper yang digunakan untuk mengantarkan course berdasarkan ID dan seluruh course.

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

   Pada StudentServiceRest

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

   Pada StudentDAO.selectAllStudents()(Diimplementasikan oleh StudentDAOImpl)

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

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

   Hasil yang diberikan dari REST API producer yang diminta akan diterima dalam bentuk objek yang sama. Saat diberikan, bentuk objek sudah sama seperti bila tanpa API.

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

   Ada dua kelas yang dibuat, yaitu CourseDAO dan CourseDAOImpl. Nantinya, StudentServiceRest akan memanggil CourseDAO, dan nantinya method CourseDAOImpl yang akan dijalankan.

Pada StudentServiceRest:

```
public class StudentServiceRest implements StudentService{

    @Autowired
    private StudentDAO studentDAO;

    @Autowired
    private CourseDAO courseDAO;

@Override
public CourseModel selectCourse(String id) {
    log.info ("REST - select course with id "+ id );
    return courseDAO.selectCourse(id);
}
@Override
public List<CourseModel> selectAllCourse() {
    log.info ("REST - select all courses");
    return courseDAO.selectAllCourse();
}
```

Jangan lupa untuk menambahkan method selectAllCourse pada StudentService.

Pada CourseDAO:

StudentDAOIm... | StudentServi... | CourseDAO.java

```
1  package com.example.dao;
2
3  import java.util.List;
4
5  import com.example.model.CourseModel;
6
7  public interface CourseDAO {
8      CourseModel selectCourse(String id);
9      List<CourseModel> selectAllCourse();
10 }
11
```

Pada 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) {
        System.out.println(id);
        CourseModel course = restTemplate.getForObject("http://localhost:8080/rest/course/view/"+id, CourseModel.class);
        return course;
    }

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

```

Bisa kita lihat bahwa implementasi tidak jauh berbeda dengan StudentDAOImpl. Bean disitu digunakan juga pada StudentDAOImpl karena saya menggunakan Spring Boot versi 1.4.0, dan disitu, bean harus diberitahu.