Hal yang Saya Pelajari pada Tutorial 7

Saya belajar membuat REST API dengan Spring pada tutorial 7 ini. Selain itu, saya belajar juga cara menangkap REST API menggunakan Spring dan mengubahnya menjadi suatu *object* di Java.

Latihan 1 : Membuat REST API untuk ViewAll Student

Saya membuat method viewAll() pada kelas StudentRestController pada *project* Tutorial07Producer. Method tersebut akan diakses dengan *endpoint* rest/student/viewall sehingga url yang diakses adalah http://localhost:8080/rest/student/viewall . Method ini akan mengembalikan *object* List of StudentModel yang didapat dari method selectAllStudents() pada *object* studentService. Tidak ada perubahan fungsionalitas pada method selectAllStudent() tersebut dari tutorial sebelumnya, Tutorial05. List of StudentModel yang dikembalikan oleh method viewAll() akan diubah menjadi JSON oleh Spring. Hal tersebut dapat terjadi karena kelas StudentRestController merupakan kelas Controller yang berperan untuk membuat REST API. Hal terebut ditandai dengan *annotation* @RestController pada kelas StudentRestController. Berikut ini *screenshot* dari method viewAll() tersebut.

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

Latihan 2 : Membuat REST API untuk ViewByIdCourse dan ViewAll Course

Saya melakukan semua *code* pada *project* Tutorial07Producer untuk Latihan 2 ini.

Saya membuat method selectAllCourse() pada *interface* CourseMapper. Method tersebut berfungsi untuk mengambil semua course yang ada di *database*. Berikut ini *screenshot* dari method tersebut :

```
@Select("select id_course as idCourse, name, credits "
        + "from course;")
List<CourseModel> selectAllCourses();
```

Saya membuat method selectAllStudentsForCourse(idCourse) pada *interface* StudentMapper. Method tersebut berfungsi untuk mengambil semua student, dari *database*, yang mengambil suatu course dengan id pada parameter course tersebut. Berikut ini *screenshot* dari method tersebut :

```
@Select("select s.npm, s.name, s.gpa "
        + "from student s, studentcourse sc "
        + "where s.npm = sc.npm and sc.id_course = #{id_course};")
List<StudentDBModel> selectAllStudentsForCourse(@Param("id_course") String idCourse);
```

Saya membuat method selectAll() pada kelas CourseService yang akan diimplementasi pada kelas *concrete*-nya yaitu CourseServiceDatabase. Berikut ini *screenshot* dari method selectAll() tersebut:

```
List<CourseModel> selectAll();
```

Selanjutnya, saya mengimplementasi method selectAll() di kelas CourseServiceDatabase. Implementasi tersebut dengan cara mengambil semua course yang ada di *database* dengan menggunakan *object* courseMapper yang bertipe CourseMapper. Setelah semua course didapatkan,

selanjutnya adalah mengambil semua mahasiswa yang mengambil course tersebut dengan bantuan method selectAllStudentForCourse(idCourse) dari *object* studentMapper yang bertipe StudentMapper. Berikut ini *screenshot* dari method selectAll() di kelas CourseServiceDatabase tersebut :

```
@Override
public List<CourseModel> selectAll() {
    List<CourseModel> courses =  courseMapper.selectAllCourses();
    if(courses != null) {
        for(CourseModel course : courses) {
            List<StudentDBModel> students = studentMapper.selectAllStudentsForCourse(course.getIdCourse());
            if(students == null) {
                students = new ArrayList<>();
            }
            course.setStudents(students);
        }
    }else {
        courses = new ArrayList<>();
    }
    return courses;
}
```

Saya membuat kelas CourseRestController yang berperan sebagai Controller untuk membuat REST API pada data Course. Pada kelas ini, terdapat dua method yaitu view(idCourse) dan viewAll(). Method view(idCourse) berfungsi untuk membuat REST API pada detail Course, hanya satu Course saja. Method tersebut akan diakses dengan *endpoint* rest/course/view/{id_course} sehingga contoh url yang dapat diakses oleh pengguna adalah http://localhost:8080/rest/course/view/CSC123 dengan CSC123 adalah id dari Course yang ingin didapatkan. Method tersebut mengembalikan *object* CourseModel yang didapatkan dari method selectCourse(idCourse) dari *object* courseService. Tidak ada perubahan fungsionalitas pada method selectCourse(idCourse) tersebut dari tutorial sebelumnya, Tutorial05.

Method selanjutnya yang saya buat pada kelas CourseRestController adalah method viewAll(). Method tersebut akan diakses dengan *endpoint* rest/course/viewall sehingga url yang diakses adalah http://localhost:8080/rest/course/viewall . Method ini akan mengembalikan *object* List of CourseModel yang didapat dari method selectAll() pada *object* courseService.

CourseModel yang dikembalikan oleh method view(idCourse) dan List of CourseModel yang dikembalikan oleh method viewAll() pada kelas CourseRestController akan diubah menjadi JSON oleh Spring. Hal tersebut dapat terjadi karena kelas CourseRestController merupakan kelas Controller yang berperan untuk membuat REST API. Hal terebut ditandai dengan *annotation* @RestController pada kelas CourseRestController. Berikut ini *screenshot* dari method view(idCourse) dan viewAll() di kelas CourseRestController tersebut :

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

    @Autowired
    CourseService courseService;

    @RequestMapping("/course/view/{idCourse}")
    public CourseModel view(@PathVariable("idCourse") String idCourse) {
        return courseService.selectCourse(idCourse);
    }

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


}
```

Latihan 3

Saya melakukan semua *code* pada *project* Tutorial07Consumer untuk Latihan 3 ini.

Saya membut method selectAllStudents() pada *interface* StudentDAO. Method tersebut akan diimplementasi pada kelas StudentDAOImpl. Berikut ini *screenshot* dari method tersebut :

```java
List < StudentModel > selectAllStudents ();
```

Saya mengimplementasi method selectAllStudents() dari *interfase* StudentDAO ke kelas StudentDAOImpl. Method ini akan mengambil REST API dari contoh url http://localhost:8080/rest/student/view/124 dengan 124 adalah npm dari student yang ingin didapatkan. REST API tersebut akan diubah menjadi *object* List of StudentModel yang akan menjadi *object* pengembalian dari method selectAllStudents() tersebut. Berikut ini *screenshot* dari method tersebut :

```java
@Override
public List < StudentModel > selectAllStudents (){
    ResponseEntity<List<StudentModel>> rateResponse = restTemplate.build()
            .exchange("http://localhost:8080/rest/student/viewall",
                    HttpMethod.GET, null, new ParameterizedTypeReference<List<StudentModel>>() {});
    return rateResponse.getBody();
}
```

Selanjutnya, saya mengimplementasi method selectAllStudents() yang ada di kelas StudentServiceRest. Method tersebut akan mengembalikan List of StudentModel yang didapatkan dari method selectAllStudents() dari *object* studentDAO. Berikut ini *screenshot* untuk method selectAllStudents() dari kelas StudentServiceRest tersebut :

```java
@Override
public List <StudentModel> selectAllStudents (){
    return studentDAO.selectAllStudents();
}
```

Latihan 4

Saya melakukan semua *code* pada *project* Tutorial07Consumer untuk Latihan 4 ini.

Saya membuat *interface* CourseDAO dengan method selectCourse(idCourse) dan selectAllCourse(). Method – method tersebut akan diimplementasi pada kelas CourseDAOImpl. Berikut ini *screenshot* dari *interface* CourseDAO tersebut :

```
public interface CourseDAO {

    CourseModel selectCourse(String idCourse);

    List<CourseModel> selectAllCourse();
}
```

Saya membuat kelas CourseDAOImpl yang mengimplementasi *interface* CourseDAO. Pada kelas ini terdapat *object* restTemplateBuilder yang bertipe RestTemplateBuilder. *Object* tersebut berfungsi untuk membuat *object* RestTemplate, dengan method build() pada kelas RestTemplateBuilder, yang berfungsi untuk mendapatkan REST API dari suatu *link* dan mengubahnya ke suatu *object* di Java. Implementasi method selectCourse(idCourse) adalah mengambil REST API, pada contoh url http://localhost:8080/rest/course/view/CSC123 dengan CSC123 adalah id dari course yang ingin diambil, dan mengubahnya menjadi *object* CourseModel dengan bantuan restTemplateBuilder.build() . Implementasi method selectAllCourse() adalah mengambil REST API, pada url http://localhost:8080/rest/course/view/viewall , dan mengubahnya menjadi *object* List of CourseModel dengan bantuan restTemplateBuilder.build(). Berikut ini *screenshot* dari kelas CourseDAOImpl tersebut :

```
@Service
public class CourseDAOImpl implements CourseDAO{

    @Autowired
    private RestTemplateBuilder restTemplateBuilder;

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

    @Override
    public List<CourseModel> selectAllCourse() {
        ResponseEntity<List<CourseModel>> rateResponse = restTemplateBuilder.build()
                .exchange("http://localhost:8080/rest/course/viewall",
                        HttpMethod.GET, null, new ParameterizedTypeReference<List<CourseModel>>() {});
        return rateResponse.getBody();
    }

}
```

Setelah itu, saya membuat kelas CourseServiceRest yang mengimplementasi *interface* CourseService. Pada kelas tersebut terdapat method selectCourse(idCourse) yang mengembalikan *object* CourseModel dari method selectCourse(idCourse) pada *object* courseDAO yang bertipe CourseDAO. Selain itu, terdapat method selectAllCourse() yang mengembalikan *object* List of CourseModel dari method selectAllCourse() pada *object* courseDAO. Berikut ini *screenshot* dari kelas tersebut :

```java
@Service
@Primary
public class CourseServiceRest implements CourseService{

    @Autowired
    private CourseDAO courseDAO;

    @Override
    public CourseModel selectCourse(String idCourse) {
        return courseDAO.selectCourse(idCourse);
    }

    @Override
    public List<CourseModel> selectAllCourse() {
        return courseDAO.selectAllCourse();
    }
}
```