1. Saya menambahkan method ini:

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

Pada StudentRestController. Method ini akan terkoneksi dengan mapper SelectAllStudents() dan menghasilkan suatu List. List ini kemudian diubah bentuknya dalam json.

2. Dikarenakan sebelumnya belum ada method untuk melihat semua course, maka saya menambahkan method viewAll pada CourseService.java

```
public interface CourseService {
        CourseModel selectCourse(String id);
        List<CourseModel> viewAll();
}
```

Kemudian mengimplementasikan method tersebut pada CourseMapper

```
@Select("select id_course, name, credits from course")
@Results(value = { @Result(property = "idCourse", column = "id_course"),
            @Result(property = "name", column = "name"), @Result(property = "credits", column = "credits"),
            @Result(property = "students", column = "id_course", javaType = List.class, many = @Many(select = "selectStudents")) })
List<CourseModel> viewAll();
```

Query tersebut mengambil semua isi table course.

```
@Override
public List<CourseModel> viewAll() {
        return courseMapper.viewAll();
}
```

Kemudian mapper tersebut dipanggil pada method yang *override* dari CourseService yang di atas.

```
@RequestMapping("/course/view/{id}")
public CourseModel viewCoursePath(@PathVariable(value = "id") String id) {
        CourseModel course = courseService.selectCourse(id);
        return course;
}
@RequestMapping("/course/viewall")
public List<CourseModel> viewAll() {
        List<CourseModel> courses = courseService.viewAll();

        return courses;
}
```

Kemudian saya membuat CourseRestController. Berisikan method viewCoursePath, yang memanggil selectCourse() dan viewAll yang memanggil viewAll() dari courseService.

Jika dijalankan akan mengeluarkan json yang sesuai.

3. Melanjutkan dari tutorial, maka untuk mengimplementasikan selectAllStudents() saya membuat method baru di StudenDAOImpl

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

Method tersebut menunjukkan restTemplate untuk menerima parameter yang akan dikirimkan oleh hurl yang dicantumkan pada method. Hasil json langsung diparse ke List.

Kemudian, pada StudenServiceRest ditambahkan method untuk memanggil selectAllStudents(). Karena ini primary, maka akan diutamakan mengambil dari method ini. Method Ini menunjuk ke arah method pada StudenDAOImpl.

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

4. Pertama saya membuat interface DAO. DI dalamnya terdapat selectCourse dan ViewAll.

```
public interface CourseDAO {
	CourseModel selectCourse(String id);
	List<CourseModel> viewAll();
}
```

Kemudian pada CourseDAOImpl, berisi method yang udah di-define pada interface. Pada selectCourse, akan menerima hasil dari /res/course/view/{id} sedangkan pada viewAll akan menerima hasil dari /rest/course/viewall

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

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

Pada CourseServiceRest, akan mengimplementasikan CourseService. CourseServiceRest merupakan primary dan akan dipilih terlebih dahulu.

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

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

5. Karena sebelumnya belum ada *controller* untuk path course/viewall, maka dibuat terlebih dahulu.

```
@RequestMapping("/course/viewall")
public String viewall(Model model) {
        List<CourseModel> courses = courseDAO.viewAll();

        model.addAttribute("courses", courses);
        return "viewall-course";
}
```

HTML untuk menampilkan seluruh course.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head th:replace="fragments/fragment :: header"></head>
<body>
      <h1>All Students</h1>
      <table id="table_id">
            <thead>
                  <tr>
                        <th>No</th>
                        <th>ID</th>
                        <th>Nama</th>
                        <th>Credits</th>
                  </tr>
            </thead>
            <tbody th:each="course,iterationStatus: ${courses}"
                  th:class="${iterationStatus.odd}?'odd'">
                  <tr>
                        <td th:text="${iterationStatus.count}">No</td>
                        <td th:text="${course.idCourse}">Student NPM</td>
                        <td th:text="${course.name}">Student Name</td>
                        <td th:text="${course.credits}">Student GPA</td>
                  </tr>
            </tbody>
      </table>
</body>
</html>
```

Kesimpulan:

Minggu ini saya mempelajari mengenai bagaimana caranya membuat REST API dan bagaimana caranya menggunakan hasil dari yang didapatkan pada API.