Ari Tri Wibowo Yudasubrata

1506735175

APAP - C

Tutorial 05

Menggunakan Database serta Relasi Database dalam Project Spring Boot

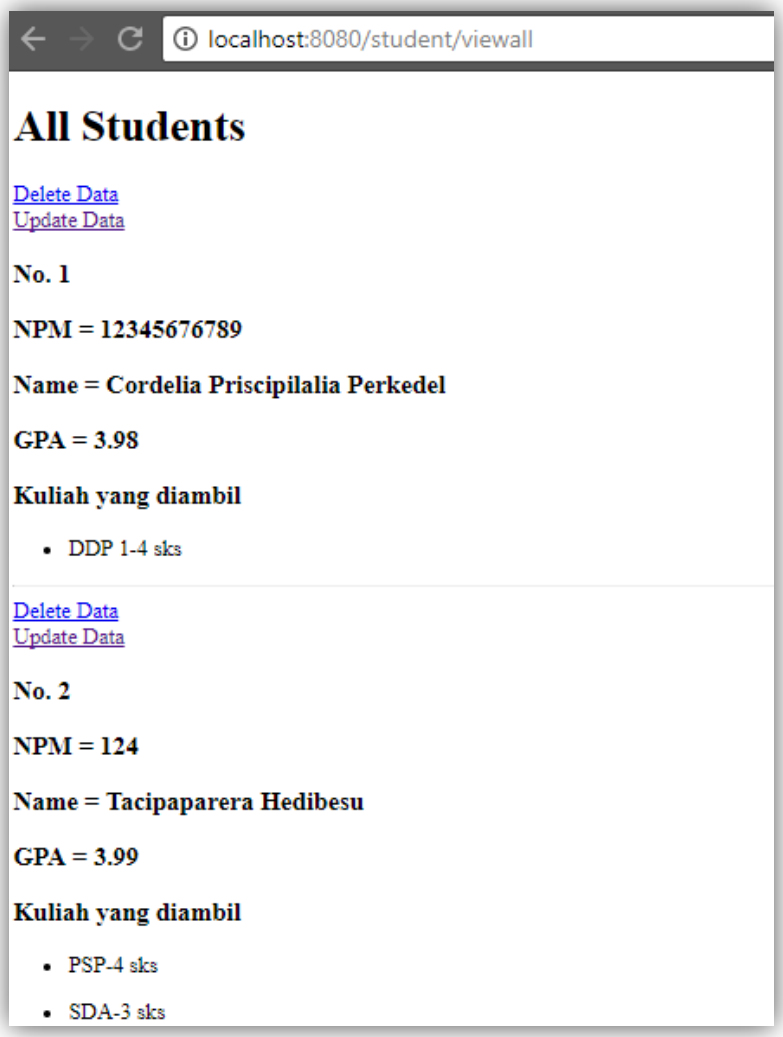
***

Yang dipelajari dalam tutorial kali ini

Lab kali ini yang dipelajari adalah cara menggunakan *database* dengan relasi dalam project Spring Boot. Di tutorial ini, terdapat dua tabel yang tersimpan dalam *database*, yakni tabel *Student* dan tabel *Course*. Kemudian akan dibuat tabel baru yakni studentcourse yang menyimpan data student dengan course yang diambilnya.

Dipelajari juga cara melakukan operasi *join* pada table *studentcourse* dengan *course* untuk menampilkan daftar kelas yang diambil oleh suatu student dengan npm tertentu dan operasi *join* pada *studentcourse* dengan *student* untuk menampilkan daftar mahasiswa yang berada di suatu kelas.

Penjelasan method untuk merubah SelectAllStudents



Method selectAllStudents pada kelas StudentMapper perlu dibuah agar menampilkan semua student beserta daftar kuliah yang diambil. Sebelumnya hanya menampilkan data student saja tidak dengan daftar kuliah yang diambil.

```
@Select("select npm, name, gpa from student")
@Results(value = {
        @Result(property = "npm", column = "npm"),
        @Result(property = "name", column = "name"),
        @Result(property = "gpa", column = "gpa"),
        @Result(property = "courses", column = "npm",
        javaType = List.class,
        many = @Many(select = "selectCourses")) })
List<StudentModel> selectAllStudents ();
```

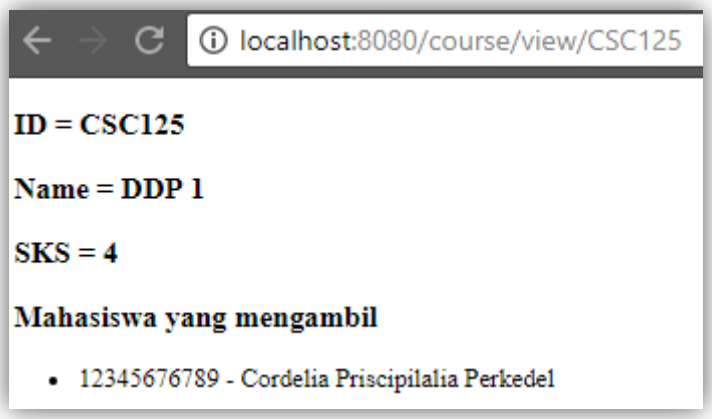
Dalam kode di atas, terdapat penampahan property courses yang menjalankan method selectCourses

```
@Result(property = "courses", column = "npm",
javaType = List.class,
many = @Many(select = "selectCourses")) })
```

Ini akan memanggil selectCourses yang berisi perintah query untuk melakukan join antara table studentcourse dengan table course sehingga didapatkan data courses-courses yang diambil oleh suatu mahasiswa

```
@Select("select course.id_course, name, credits " +
        "from studentcourse join course " +
        "on studentcourse.id_course = course.id_course " +
        "where studentcourse.npm = #{npm}")
List<CourseModel> selectCourses (@Param("npm") String npm);
```

Penjelasan method untuk menambahkan view pada course



View course/view{id} ditambahkan agar dapat menampikan informasi data course berupa ID, nama, SKS, dan daftar mahasiswa yang diambil.

Membuat class CourseModel yang menyimpan data course beserta List yang berisi students yang mengambil kelas tersebut

```java
public class CourseModel {

    private String idCourse;
    private String name;
    private Integer credits;
    private List<StudentModel> students;

}
```

Kemudian membuat interface CourseService dengan method selectCourse

```java
import com.example.model.CourseModel;

public interface CourseService
{

    CourseModel selectCourse (String idcourse);

}
```

Lalu Membuat class interface courseMapper

Pertama-tama definisikan variable selectStudentCourses yang melakukan join antara table student dengan table studentcourses, sehingga didapatkan data mahasiswa dalam suatu course yang kemudian akan disimpan dalam sebuah List.

Kemudian List selectStudentCOurses ini, aka dipakai dalam sebuah object CourseModel bernama selectCoursewithStudent yang menyimpan idCourse, name, credits, dan daftar mahasiswa di dalamnya

```java
public interface CourseMapper
{

    @Select("select id_course, name, credits from course where id_course = #{id}")
    @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="selectStudentCourses"))
    })
    CourseModel selectCoursewithStudent (@Param("id") String id);

    @Select("select name, student.npm " +
            "from studentcourse join student " +
            "on studentcourse.npm = student.npm " +
            "where studentcourse.id_course = #{id}")
    List<StudentModel> selectStudentCourses (@Param("id") String id);

}
```

Kemudian membuat class CourseServiceDatabase yang mengimport CourseMapper

```java
public class CourseServiceDatabase implements CourseService
{
    @Autowired
    private CourseMapper courseMapper;

    @Override
    public CourseModel selectCourse (String idCourse)
    {
        log.info ("select course with id {}", idCourse);
        return courseMapper.selectCoursewithStudent(idCourse);
    }

}
```

Kemudian membuat class CourseController yang salah satu fungsinya yakni menghandle ketika mengakses url /course/view/{id}

```java
public class CourseController
{
    @Autowired
    CourseService courseDAO;

    /*
    @RequestMapping("/")
    public String index ()
    {
        return "index";
    }
    */

    @RequestMapping("/course/view/{id}")
    public String viewCourse (Model model,
            @PathVariable(value = "id") String idcourse) {

        CourseModel course = courseDAO.selectCourse (idcourse);
        if (course != null) {
            model.addAttribute ("course", course);
            return "viewforcourse";
        } else {
            model.addAttribute ("id", idcourse);
            return "not-found";
        }

    }

}
```