mybatis resultMap #2

2021-10-17

이승진

**학습목표**

mybatis resultmap 기능을 학습한다

**목차**

[1. 개요 2](#_Toc54938094)

[1) 교수별 강좌목록 2](#_Toc54938095)

[2) DTO 클래스 구현 방법 #1 3](#_Toc54938096)

[3) DTO 클래스 구현 방법 #2 4](#_Toc54938097)

[2. 교수별 강좌 목록 구현 #1 5](#_Toc54938098)

[1) 프로젝트 생성 5](#_Toc54938099)

[2) pom.xml 7](#_Toc54938100)

[3) application.properties 7](#_Toc54938101)

[4) Course.java 8](#_Toc54938102)

[5) Professor.java 8](#_Toc54938103)

[6) CourseMapper.java 9](#_Toc54938104)

[7) ProfessorMapper.java 9](#_Toc54938105)

[8) ProfessorController.java 10](#_Toc54938106)

[9) common.css 11](#_Toc54938107)

[10) professor/list.jsp 12](#_Toc54938108)

[3. 교수별 강좌 목록 구현 #2 13](#_Toc54938109)

[1) SQL 13](#_Toc54938110)

[2) 프로젝트 생성 14](#_Toc54938111)

[3) pom.xml 16](#_Toc54938112)

[4) application.properties 16](#_Toc54938113)

[5) Course.java 17](#_Toc54938114)

[6) Professor.java 17](#_Toc54938115)

[7) ProfessorMapper.java 18](#_Toc54938116)

[8) ProfessorMapper.xml 18](#_Toc54938117)

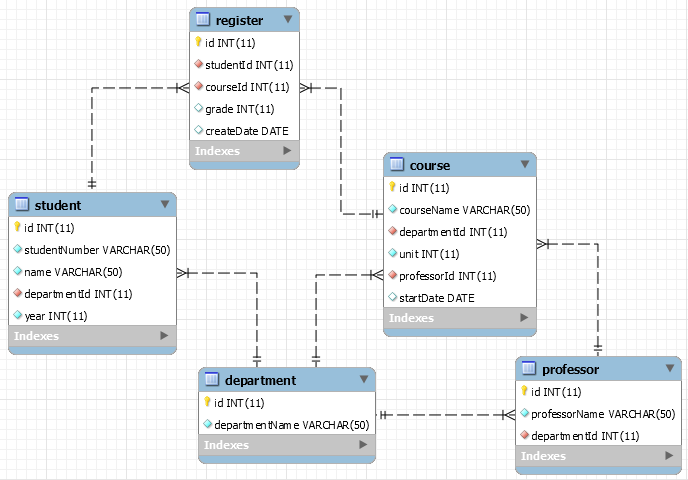
[9) ProfessorController.java 20](#_Toc54938118)

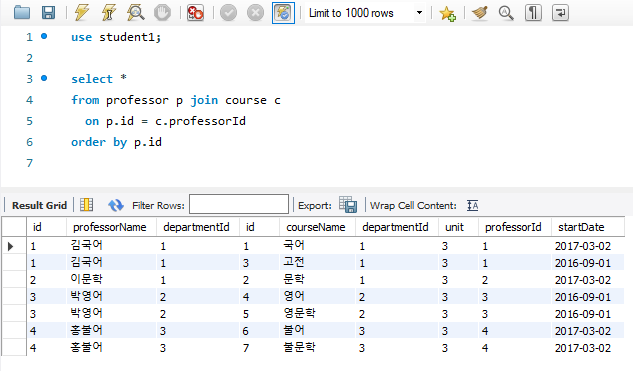
[10) common.css 21](#_Toc54938119)

[11) professor/list.jsp 22](#_Toc54938120)

# 개요

## 교수별 강좌목록





|  |
| --- |
| select \*  from professor p join course c  on p.id = c.professorId  order by p.id |

이 명령의 조회 결과 출력 화면을 구현하자.

위 조회 결과의 컬럼은 다음과 같다.

|  |  |
| --- | --- |
| id | professor 테이블 |
| professorName | professor 테이블 |
| departmentId | professor 테이블 |
| id | course 테이블 |
| courseName | course 테이블 |
| departmentId | course 테이블 |
| unit | course 테이블 |
| professorId | course 테이블 |
| startDate | course 테이블 |

## DTO 클래스 구현 방법 #1

만약 조회 결과 레코드 한 개를 어떤 DTO 객체 한 개에 다 채워야 한다면,

Course DTO 클래스에 professorName, professorDepartmentId 속성을 추가하여,

이 DTO 클래스에 채우면 된다.

|  |
| --- |
| @Data  public class Course {  int id;  String courseName;  int departmentId;  int unit;  int professorId;  Date startDate;  String professorName;  int professorDepartmentId;  } |

이 Course 클래스의 속성들중, course 테이블의 필드에 해당하는 속성은 6 개이고,

professor 테이블의 필드에 해당하는 속성은 2 개이다.

그런데, 이런 방식으로 구현을 하다보면,

Course DTO 클래스의 속성들 중, course 테이블의 필드에 해당하는 속성보다

다른 테이블의 필드에 해당하는 속성이 더 많아지는 상황까지 발전하게 된다.

DTO 클래스 구조가 지저분해진다.

## DTO 클래스 구현 방법 #2

course 테이블의 필드는 Course DTO 객체에 채우고,

professor 테이블의 필드는 Professor DTO 객체에 채우고,

이 DTO 객체들을 서로 연결하여 구현하는 방법이 더 객체지향적이다

### association

만약 Course DTO 객체에 professor 테이블의 필드도 채우려면,

Course DTO 클래스를 다음과 같이 구현한다.

|  |
| --- |
| @Data  public class Course {  int id;  String courseName;  int departmentId;  int unit;  int professorId;  Date startDate;  Professor professor;  } |

Course 하나를 담당하는 Professor는 한 명이다.

Course DTO 클래스의 Professor 속성을 association 속성이라고 부르자.

이 association 속성에 데이터를 채우는 구현 방법은,

지난 시간의 resultMap 예제의 구현 방법과 같다.

### collection

만약 Professor DTO 객체에 course 테이블의 필드도 채우려면,

Professor DTO 클래스를 다음과 같이 구현한다.

|  |
| --- |
| @Data  public class Professor {  int id;  String professorName;  int departmentId;  List<Course> courses;  } |

Professor 한 명이 담당하는 Course는 여러 개이다.

Professor DTO 클래스의 List<Course> 속성을 collection 속성이라고 부르자.

이 collection 속성에 데이터를 채우는 구현 방법을 살펴보자.

이 구현 방법은 두 개이다.

(1) 먼저 교수 목록을 조회하고,

교수 각각에 대해서 담당 강좌 목록을 조회하여 교수 DTO 객체에 채운다.

구현은 비교적 단순하지만, 여러 번 조회해야 하는 것이 단점이다.

|  |
| --- |
| ProfessorMapper  @Select("SELECT \* FROM professor")  List<Professor> findAll();    CourseMapper  @Select("SELECT \* FROM course WHERE professorId = #{professorId}")  List<Course> findByProfessorId(int professorId);  List<Professor> professors = professorMapper.findAll();  for (Professor p : professors) {  List<Courses> courses = courseMapper.findByProfessorId(p.getId())  p.setCourses(courses);  } |

(2) 교수별 담당 강좌 목록 전체를 한 번에 조회해서

교수 DTO 객체와 강좌 DTO 객체에 나누어 채운다. (위에서 구현한 Professor DTO 클래스에 채운다)

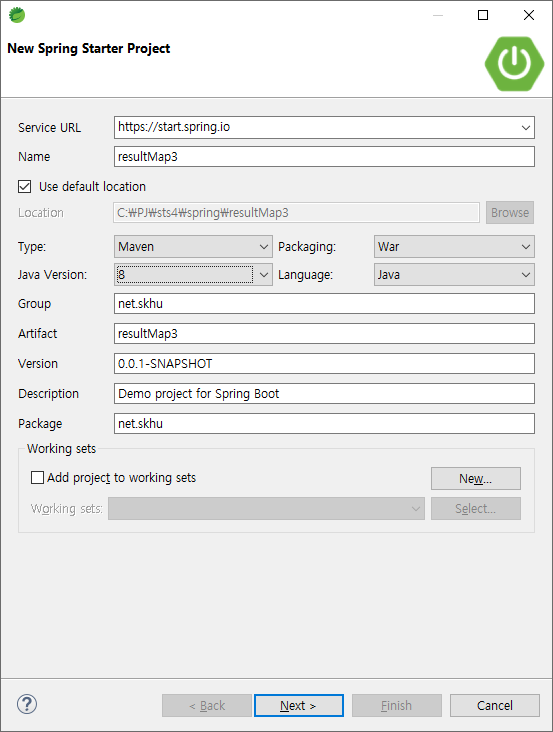
이렇게 나누어 채우는 방법을 mybatis resultMap 기능으로 구현해야 한다.

한 번만 조회하면 된다.

|  |
| --- |
| select \*  from professor p join course c  on p.id = c.professorId |

# 교수별 강좌 목록 구현 #1

## 프로젝트 생성



|  |  |
| --- | --- |
| Name | resultMap3 |
| Package | net.skhu |

Graphical user interface, text, application

Description automatically generated

## pom.xml

|  |
| --- |
| <dependency>  <groupId>org.apache.tomcat.embed</groupId>  <artifactId>tomcat-embed-jasper</artifactId>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  </dependency> |

## application.properties

|  |
| --- |
| spring.mvc.view.prefix=/WEB-INF/views/  spring.mvc.view.suffix=.jsp  spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  spring.datasource.url=jdbc:mysql://localhost:3306/student1?useUnicode=yes&characterEncoding=UTF-8&allowMultiQueries=true&serverTimezone=Asia/Seoul  spring.datasource.username=user1  spring.datasource.password=skhuA+4.5  server.port=8088 |

## Course.java

src/main/java/net/skhu/dto/Course.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | package net.skhu.dto;  import java.util.Date;  import lombok.Data;  @Data  public class Course {  int id;  String courseName;  int departmentId;  int unit;  int professorId;  Date startDate;  } |

## Professor.java

src/main/java/net/skhu/dto/Professor.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | package net.skhu.dto;  import java.util.List;  import lombok.Data;  @Data  public class Professor {  int id;  String professorName;  int departmentId;  List<Course> courses;  } |

## CourseMapper.java

src/main/java/net/skhu/mapper/CourseMapper.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | package net.skhu.mapper;  import java.util.List;  import org.apache.ibatis.annotations.Mapper;  import org.apache.ibatis.annotations.Select;  import net.skhu.dto.Course;  @Mapper  public interface CourseMapper {  @Select("SELECT \* FROM course WHERE professorId = #{professorId}")  List<Course> findByProfessorId(int professorId);  } |

## ProfessorMapper.java

src/main/java/net/skhu/mapper/ProfessorMapper.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | package net.skhu.mapper;  import java.util.List;  import org.apache.ibatis.annotations.Mapper;  import org.apache.ibatis.annotations.Select;  import net.skhu.dto.Professor;  @Mapper  public interface ProfessorMapper {  @Select("SELECT \* FROM professor")  List<Professor> findAll();  } |

## ProfessorController.java

src/main/net/skhu/controller/ProfessorController.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | package net.skhu.controller;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.RequestMapping;  import net.skhu.dto.Professor;  import net.skhu.mapper.CourseMapper;  import net.skhu.mapper.ProfessorMapper;  @Controller  public class ProfessorController {  @Autowired ProfessorMapper professorMapper;  @Autowired CourseMapper courseMapper;  @RequestMapping("professor/list")  public String list(Model model) {  List<Professor> professors = professorMapper.findAll();  for (Professor professor : professors)  professor.setCourses(courseMapper.findByProfessorId(professor.getId()));  model.addAttribute("professors", professors);  return "professor/list";  }  } |

List<Professor> professors = professorMapper.findAll();

먼저 교수 목록을 조회한다.

for (Professor professor : professors)

professor.setCourses(courseMapper.findByProfessorId(professor.getId()));

교수 목록의 교수 각각에 대해서,

그 교수가 담당하는 강좌 목록을 조회하여, 교수 DTO 객체에 채운다.

## common.css

src/main/resources/static/common.css

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | div.container { width: 800px; margin: 10px auto; font-size: 10pt; }  .btn { padding: 0.4em 1em; border: 1px solid gray;  border-radius: 0.5em; background: linear-gradient(#fff, #ddd);  text-decoration: none; color: black;  display: inline-block; }  .btn:active {  -ms-transform: translateY(2px);  -webkit-transform: translateY(2px);  transform: translateY(2px);  background: #ccc; }  table.list { border-collapse: collapse; width: 100%; }  table.list td { padding: 4px; border: 1px solid gray; }  table.list th { padding: 4px; border: 1px solid gray; background-color: #eee; }  input { padding: 4px; }  select { padding: 4px; }  tr[data-url]:hover { cursor: pointer; background-color: #ffe; } |

## professor/list.jsp

src/main/webapp/WEB-INF/views/professor/list.jsp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>  <script src="/common.js"></script>  <link rel="stylesheet" type="text/css" href="/common.css" />  <style>  h3 { margin-bottom: 2px; }  td:nth-child(1), td:nth-child(5) { text-align: center; }  </style>  </head>  <body>  <div class="container">  <h1>교수별 담당강좌</h1>  <c:forEach var="professor" items="${ professors }">  <h3>${ professor.id } - ${ professor.professorName }</h3>  <table class="list">  <thead>  <tr>  <th>id</th>  <th>강좌명</th>  <th>학점</th>  <th>시작일</th>  </tr>  </thead>  <tbody>  <c:forEach var="course" items="${ professor.courses }">  <tr>  <td>${ course.id }</td>  <td>${ course.courseName }</td>  <td>${ course.unit }</td>  <td><fmt:formatDate pattern="yyyy-MM-dd" value="${ course.startDate }" /></td>  </tr>  </c:forEach>  </tbody>  </table>  </c:forEach>  </div>  </body>  </html> |

## 실행

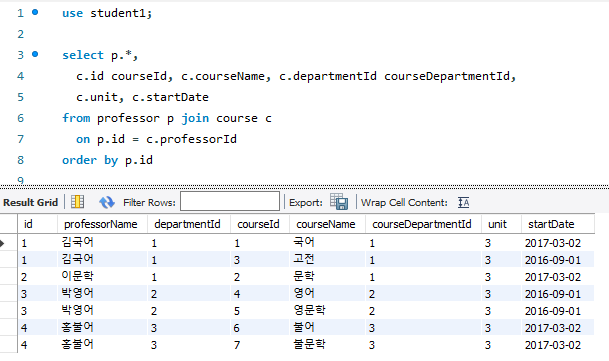
http://localhost:8088/professor/list

Table

Description automatically generated with medium confidence

# 교수별 강좌 목록 구현 #2

## SQL

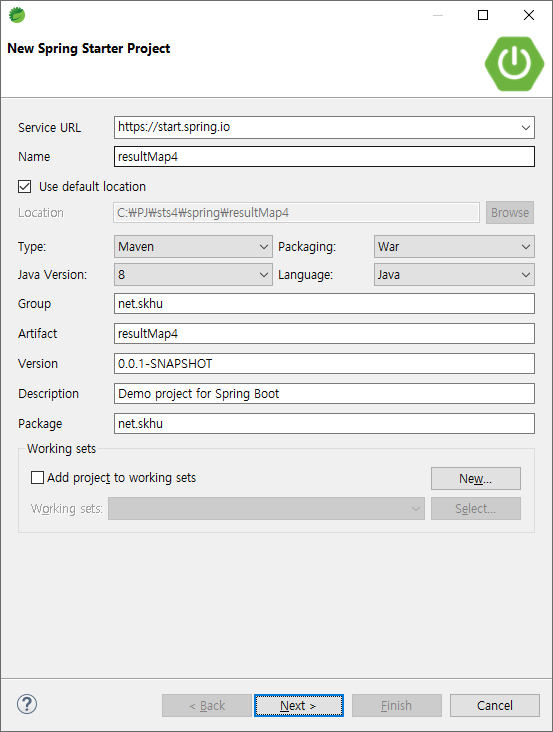


위 SQL 명령을 한 번 실행해서, 교수 목록과 교수가 담당하는 강좌 목록을 조회한다.

조회 결과에 컬럼명이 중복되지 않도록 컬렴명을 조정하였다.

|  |
| --- |
| select p.\*,  c.id courseId, c.courseName, c.departmentId courseDepartmentId,  c.unit, c.startDate  from professor p join course c  on p.id = c.professorId  order by p.id |

## 프로젝트 생성



|  |  |
| --- | --- |
| Name | resultMap4 |
| Package | net.skhu |

Graphical user interface, text, application

Description automatically generated

## pom.xml

|  |
| --- |
| <dependency>  <groupId>org.apache.tomcat.embed</groupId>  <artifactId>tomcat-embed-jasper</artifactId>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  </dependency> |

## application.properties

|  |
| --- |
| spring.mvc.view.prefix=/WEB-INF/views/  spring.mvc.view.suffix=.jsp  spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  spring.datasource.url=jdbc:mysql://localhost:3306/student1?useUnicode=yes&characterEncoding=UTF-8&allowMultiQueries=true&serverTimezone=Asia/Seoul  spring.datasource.username=user1  spring.datasource.password=skhuA+4.5  server.port=8088 |

## Course.java

src/main/java/net/skhu/dto/Course.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | package net.skhu.dto;  import java.util.Date;  import lombok.Data;  @Data  public class Course {  int id;  String courseName;  int departmentId;  int unit;  int professorId;  Date startDate;  } |

이 파일은 앞의 예제와 동일하다.

## Professor.java

src/main/java/net/skhu/dto/Professor.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | package net.skhu.dto;  import java.util.List;  import lombok.Data;  @Data  public class Professor {  int id;  String professorName;  int departmentId;  List<Course> courses;  } |

이 파일은 앞의 예제와 동일하다.

## ProfessorMapper.java

src/main/java/net/skhu/mapper/ProfessorMapper.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | package net.skhu.mapper;  import java.util.List;  import org.apache.ibatis.annotations.Mapper;  import net.skhu.dto.Professor;  @Mapper  public interface ProfessorMapper {  List<Professor> findAll();  } |

## ProfessorMapper.xml

src/main/java/net/skhu/mapper/ProfessorMapper.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | <?xml version="1.0" encoding="UTF-8" ?>  <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  <mapper namespace="net.skhu.mapper.ProfessorMapper">  <select id="findAll" resultMap="professorWithCourses">  select p.\*,  c.id courseId, c.courseName, c.departmentId courseDepartmentId,  c.unit, c.startDate  from professor p join course c  on p.id = c.professorId  order by p.id  </select>  <resultMap id="professorWithCourses" type="net.skhu.dto.Professor">  <id property="id" column="id" />  <result property="professorName" column="professorName" />  <result property="departmentId" column="departmentId" />  <collection property="courses" ofType="net.skhu.dto.Course">  <id property="id" column="courseId" />  <result property="courseName" column="courseName" />  <result property="departmentId" column="courseDepartmentId" />  <result property="unit" column="unit" />  <result property="startDate" column="startDate" />  </collection>  </resultMap>  </mapper> |

<id property="id" column="id" />

조회 결과의 id 컬럼은 Professor 클래스의 id 속성(property)에 채운다.

이 컬럼이 professor 테이블의 기본키(primary key)이기 때문에 <id> 태그를 사용하였다.

<result property="professorName" column="professorName" />

조회 결과의 professorName 컬럼은 Professor 클래스의 professorName 속성(property)에 채운다.

이 컬럼이 professor 테이블의 기본키가 아니기 때문에 <result> 태그를 사용하였다.

<collection property="courses" ofType="net.skhu.dto.Course">

Professor 클래스의 courses 속성에 Course 객체 목록이 채워진다.

Professor 클래스의 courses 속성은 List<Course> 타입이다.

이렇게 목록(List) 형태의 속성에 값을 채울 때 <collection> 태그를 사용한다.

<id property="id" column="courseId" />

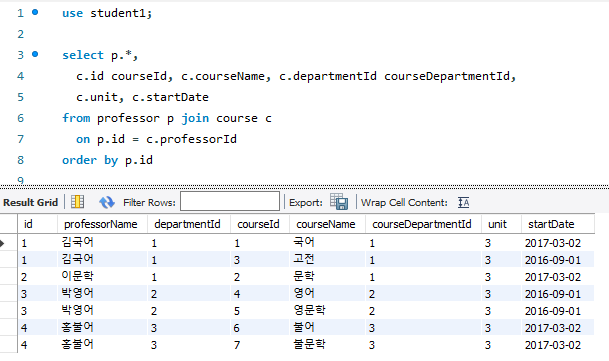
조회 결과의 courseId 컬럼은 Course 클래스의 id 속성에 채운다.

이 컬럼이 course 테이블의 기본키이기 때문에 <id> 태그를 사용하였다.

<result property="courseName" column="courseName"/>

조회 결과의 courseName 컬럼은 Course 클래스의 courseName 속성(property)에 채운다.

이 컬럼이 course 테이블의 기본키가 아니기 때문에 <result> 태그를 사용하였다.



위 조회 결과는 mybatis resultMap 에 의해서 다음과 같은 객체 구조로 형태로 변형된다.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Professor 객체 | | | | | | | |
| id | professorName | departmentId | List<Course> courses | | | | |
|  | Course 객체 | | | | |
|  | id | courseName | departmentId | unit | startDate |
| 1 | 김국어 | 1 | 1 | 국어 | 1 | 3 | 2017-03-02 |
| 3 | 고전 | 1 | 3 | 2016-09-01 |
| 2 | 이문학 | 1 | 2 | 문학 | 1 | 3 | 2017-03-02 |
| 3 | 박영어 | 2 | 4 | 영어 | 2 | 3 | 2016-09-01 |
| 5 | 영문학 | 2 | 3 | 2016-09-01 |
| 4 | 홍불어 | 3 | 6 | 불어 | 3 | 3 | 2017-03-02 |
| 7 | 불문학 | 3 | 3 | 2017-03-02 |

김국어 Professor 객체의 courses 속성에는 2 개의 Course 객체 목록(List<Course>)이 채워진다.

이문학 Professor 객체의 courses 속성에는 1 개의 Course 객체 목록(List<Course>)이 채워진다.

박영어 Professor 객체의 courses 속성에는 2 개의 Course 객체 목록(List<Course>)이 채워진다.

홍불어 Professor 객체의 courses 속성에는 2 개의 Course 객체 목록(List<Course>)이 채워진다.

위 과정은 myBatis의 resultMap 기능에 의해서 자동으로 실행된다.

## ProfessorController.java

src/main/net/skhu/controller/ProfessorController.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | package net.skhu.controller;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.RequestMapping;  import net.skhu.mapper.ProfessorMapper;  @Controller  public class ProfessorController {  @Autowired ProfessorMapper professorMapper;  @RequestMapping("professor/list")  public String list(Model model) {  model.addAttribute("professors", professorMapper.findAll());  return "professor/list";  }  } |

## common.css

src/main/resources/static/common.css

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | div.container { width: 800px; margin: 10px auto; font-size: 10pt; }  .btn { padding: 0.4em 1em; border: 1px solid gray;  border-radius: 0.5em; background: linear-gradient(#fff, #ddd);  text-decoration: none; color: black;  display: inline-block; }  .btn:active {  -ms-transform: translateY(2px);  -webkit-transform: translateY(2px);  transform: translateY(2px);  background: #ccc; }  table.list { border-collapse: collapse; width: 100%; }  table.list td { padding: 4px; border: 1px solid gray; }  table.list th { padding: 4px; border: 1px solid gray; background-color: #eee; }  input { padding: 4px; }  select { padding: 4px; }  tr[data-url]:hover { cursor: pointer; background-color: #ffe; } |

이 파일은 앞의 예제와 동일하다.

## professor/list.jsp

src/main/webapp/WEB-INF/views/professor/list.jsp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>  <script src="/common.js"></script>  <link rel="stylesheet" type="text/css" href="/common.css" />  <style>  h3 { margin-bottom: 2px; }  td:nth-child(1), td:nth-child(5) { text-align: center; }  </style>  </head>  <body>  <div class="container">  <h1>교수별 담당강좌</h1>  <c:forEach var="professor" items="${ professors }">  <h3>${ professor.id } - ${ professor.professorName }</h3>  <table class="list">  <thead>  <tr>  <th>id</th>  <th>강좌명</th>  <th>학점</th>  <th>시작일</th>  </tr>  </thead>  <tbody>  <c:forEach var="course" items="${ professor.courses }">  <tr>  <td>${ course.id }</td>  <td>${ course.courseName }</td>  <td>${ course.unit }</td>  <td><fmt:formatDate pattern="yyyy-MM-dd" value="${ course.startDate }" /></td>  </tr>  </c:forEach>  </tbody>  </table>  </c:forEach>  </div>  </body>  </html> |

이 파일은 앞의 예제와 동일하다.

## 실행

http://localhost:8088/professor/list

Table

Description automatically generated

## 연습문제

resultmap 기능을 이용하여

학과별 소속 학생 출력 기능을 구현하라.

|  |
| --- |
| @Data  class Department {  int id;  String departmentName;  List<Student> students;  } |