**데이터 목록 조회**

2021-10-29

이승진

**학습목표**

데이터베이스 연결 정보 설정 방법

데이터 목록을 조회하여 화면에 출력하는 기능을 구현한다.

mybatis 데이터베이스 프로그래밍 기술

**목차**

[1. 개요 2](#_Toc86367487)

[2. mybatis1 프로젝트 2](#_Toc86367488)

[1) 프로젝트 생성 2](#_Toc86367489)

[2) pom.xml 파일 수정 5](#_Toc86367490)

[3) application.properties 파일 수정 6](#_Toc86367491)

[4) student 테이블 7](#_Toc86367492)

[5) Student.java 생성 8](#_Toc86367493)

[6) StudentMapper.java 생성 11](#_Toc86367494)

[7) StudentController.java 생성 16](#_Toc86367495)

[8) student/list.jsp 생성 17](#_Toc86367496)

[9) 실행 18](#_Toc86367497)

[3. 실습 19](#_Toc86367498)

# 개요

mybatis 프레임웍을 이용한 데이터베이스 프로그래밍 기술을 배운다.

Java의 기초 데이터베이스 프로그래밍 기술은 JDBC 이다.

JDBC 는 너무 기초라서 코딩할 양이 매우 많고 불편하다.

# mybatis1 프로젝트

## 프로젝트 생성

메뉴: File - New - Spring Starter Project

Graphical user interface, text, application, email

Description automatically generated

|  |  |
| --- | --- |
| Name | mybatis1 |
| Type | Maven |
| Packaing | War |
| Group | net.skhu |
| Artifact | mybatis1 |
| Package | net.skhu |

Graphical user interface, text, application

Description automatically generated

Developer Tools 항목 아래에서

Spring Boot DevTools 항목을 선택한다.

SQL 항목 아래에서

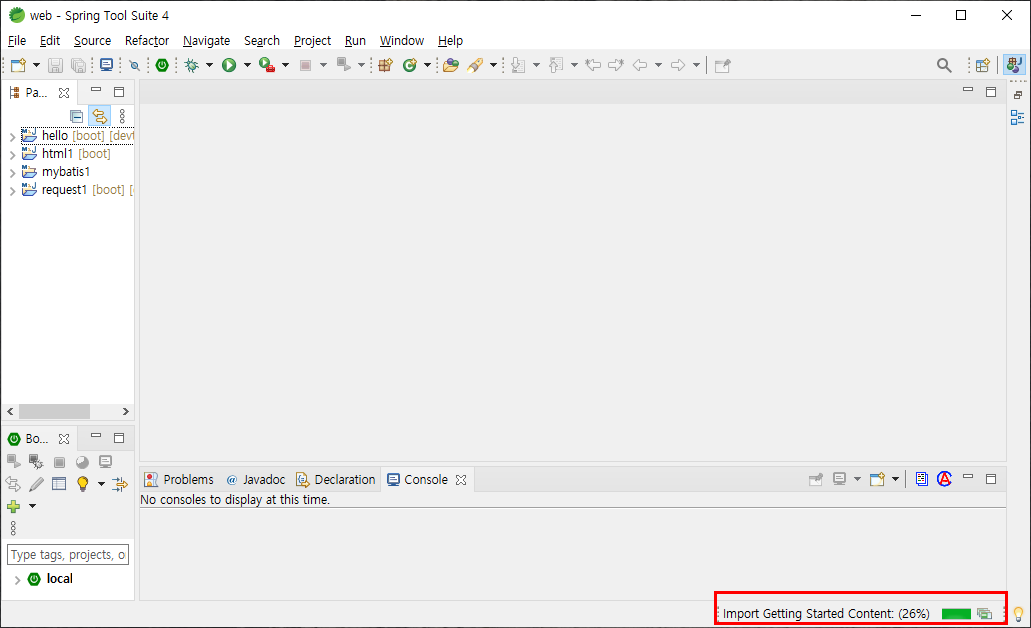
MyBatis Framework 항목과

MySQL Driver 항목을 선택한다.

Web 항목 아래에서

Spring Web 항목을 선택한다.

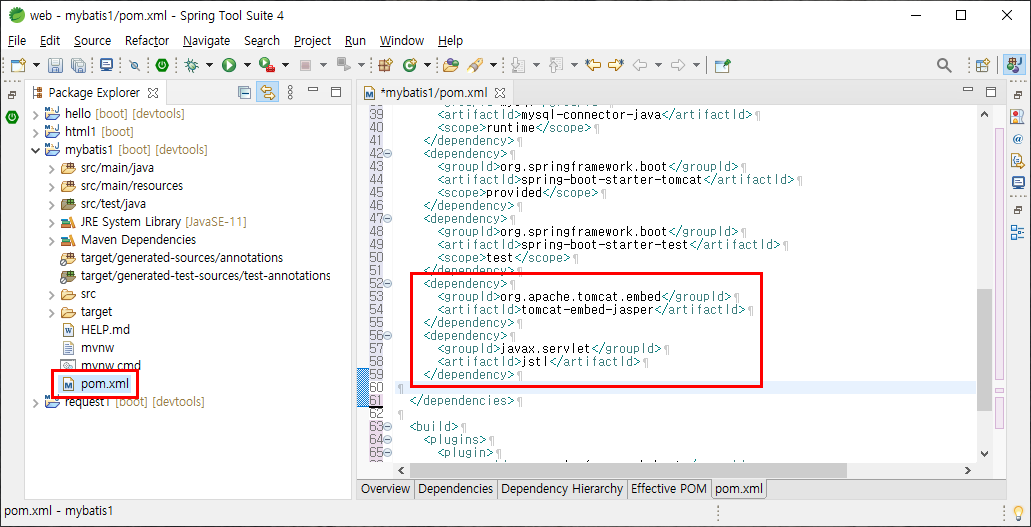
Finish 버튼 클릭



프로젝트 생성 완료까지 조금 기다려야 한다.

STS에서 어떤 작업이 진행중일 때, 화면 오른쪽 아래에 작업 진척도가 표시된다.

## pom.xml 파일 수정



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

위 내용을 </dependencies> 태그 바로 위에 삽입하자.

spring 프로젝트를 만들 때, 언제나 위와 같이 수정해야 한다.

## application.properties 파일 수정

Graphical user interface, text, application

Description automatically generated

|  |
| --- |
| 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/student2?useUnicode=yes&characterEncoding=UTF-8&allowMultiQueries=true&serverTimezone=Asia/Seoul  spring.datasource.username=user1  spring.datasource.password=skhuA+4.5  server.port=8088 |

spring.mvc.view.prefix=/WEB-INF/views/

spring.mvc.view.suffix=.jsp

뷰 파일(JSP)의 위치를 지정한다.

spring.datasource.url=jdbc:mysql://localhost:3306/student2?useUnicode=yes&characterEncoding=UTF-8&allowMultiQueries=true&serverTimezone=Asia/Seoul

이 부분은 강의노트 폭이 좁아서 두 줄로 표시되었다. 한 줄로 입력해야 한다.

MySQL 서버 주소와 데이터베이스 이름을 지정한다.

내 MySQL이 내 PC에 설치되어 있기 때문에 주소가 localhost 이다.

연결할 데이터베이스 이름은 student2 이다.

spring.datasource.username=user1

spring.datasource.password=skhuA+4.5

MySQL을 설치할 때 생성한 계정 이름과 비밀번호를 입력한다.

server.port=8088

spring 프로젝트에 내장된 톰캣 서버가 사용할 포트 번호를 지정한다.

## student 테이블

Table

Description automatically generated with medium confidence

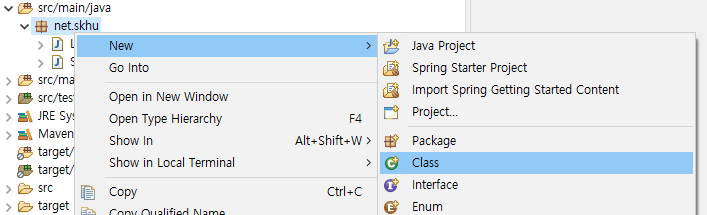
Graphical user interface

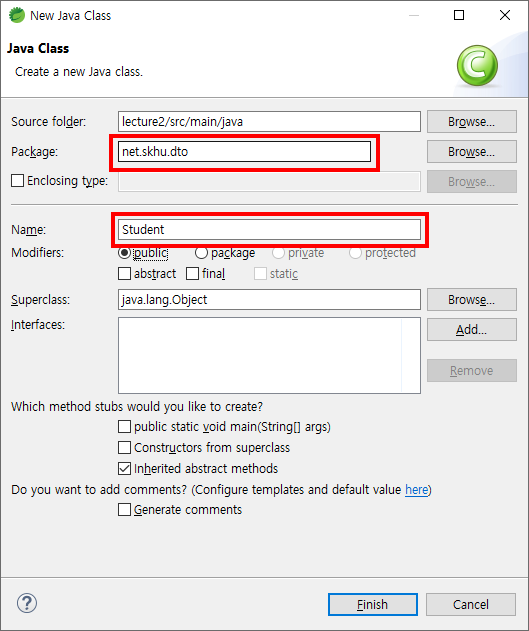
Description automatically generated

student2 데이터베이스의 student 테이블의 내용은 위와 같다.

student 테이블에서 조회한 데이터를 담기 위한 Student 클래스를 만들자.

## Student.java 생성





|  |  |
| --- | --- |
| Package | net.skhu.dto |
| Name | Student |

### src/main/java/net/skhu/dto/Student.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  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77 | package net.skhu.dto;  public class Student {  int id;  String studentNo;  String name;  int departmentId;  String phone;  String sex;  String email;  String departmentName;  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public String getStudentNo() {  return studentNo;  }  public void setStudentNo(String studentNo) {  this.studentNo = studentNo;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public int getDepartmentId() {  return departmentId;  }  public void setDepartmentId(int departmentId) {  this.departmentId = departmentId;  }  public String getPhone() {  return phone;  }  public void setPhone(String phone) {  this.phone = phone;  }  public String getSex() {  return sex;  }  public void setSex(String sex) {  this.sex = sex;  }  public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }  public String getDepartmentName() {  return departmentName;  }  public void setDepartmentName(String departmentName) {  this.departmentName = departmentName;  }  } |

student 테이블에서 조회한 데이터를 채워서 전달하기 위한 객체이다.

이런 객체를 DTO (Data Transfer Object) 라고 부른다.

int id;

String studentNo;

String name;

int departmentId;

String phone;

String sex;

String email;

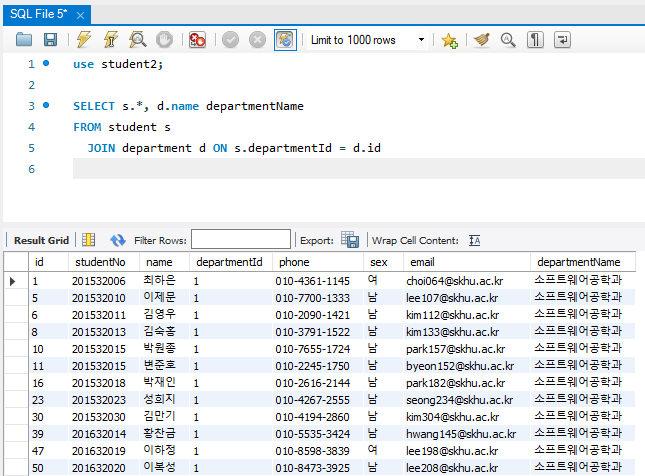
위 멤버 변수들은 student 테이블의 필드와 일치한다.

String departmentName;

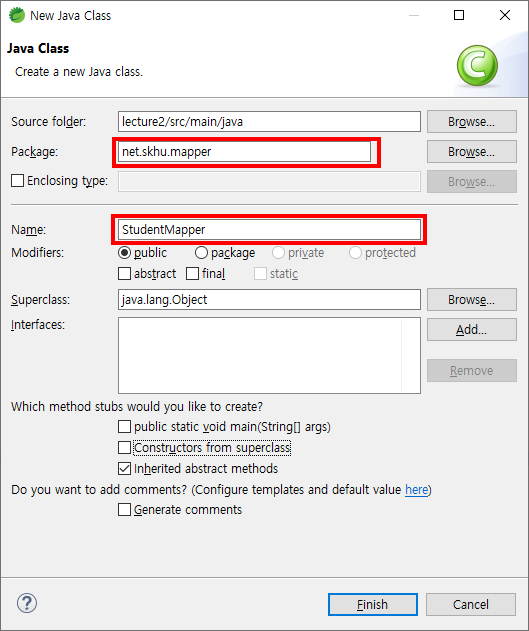
student 테이블에는 departmentName 필드가 없지만,

student 테이블을 조회할 때, department 테이블과 조인해서 학과명도 같이 조회한다.

그래서 Student DTO 클래스에 departmentName 멤버 변수를 추가하였다.



## StudentMapper.java 생성



### src/main/java/net/skhu/mapper/StudentMapper.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | package net.skhu.mapper;  import java.util.List;  import org.apache.ibatis.annotations.Mapper;  import org.apache.ibatis.annotations.Select;  import net.skhu.dto.Student;  @Mapper  public interface StudentMapper {  @Select("SELECT s.\*, d.name departmentName " +  "FROM student s JOIN department d ON s.departmentId = d.id")  List<Student> findAll();  } |

StudentMapper는 DB의 student 테이블에 대한 SQL 명령들을 구현하기 위한 Java interface 이다.

클래스가 아니고 Java interface임에 주목하자.

StudentMapper와 같은 것들을 MyBatis mapper 라고 부른다.

### @Mapper

mybatis mapper에는 @Mapper 어노테이션이 붙어있어야 한다.

### findAll

|  |
| --- |
| @Select("SELECT s.\*, d.name departmentName " +  "FROM student s JOIN department d ON s.departmentId = d.id")  List<Student> findAll(); |

findAll 메소드는, student 테이블의 모든 레코드를 조회해서

조회 결과 레코드 각각을 Student 객체에 채우고

그 Student 객체들을 List<Student> 객체에 채워서 리턴한다.

findAll 메소드를 호출하면,

SELECT s.\*, d.name departmentName FROM student s JOIN department d ON s.departmentId = d.id

SQL 명령이 MySQL 엔진에서 실행된다.

@Select("SELECT s.\*, d.name departmentName " +

"FROM student s JOIN department d ON s.departmentId = d.id")

SQL 명령이 너무 길기 때문에 두 줄로 나누어 입력했다.

"SELECT s.\*, d.name departmentName " 문자열과

"FROM student s JOIN department d ON s.departmentId = d.id" 문자열이 더해져서

아래와 같은 긴 문자열이 된다.

"SELECT s.\*, d.name departmentName FROM student s JOIN department d ON s.departmentId = d.id"

분홍색으로 칠한 공백 문자가 반드시 필요하다.

이 공백 문자가 없다면 SQL 문법 오류가 발생할 것이다.

Graphical user interface, text, application

Description automatically generated

FROM 앞에 공백이 없으면 위와 같은 SQL 문법 오류가 발생한다.

|  |
| --- |
| @Select("SELECT s.\*, d.name departmentName " +  "FROM student s JOIN department d ON s.departmentId = d.id")  List<Student> findAll(); |

SELECT s.\*, d.name departmentName FROM student s JOIN department d ON s.departmentId = d.id

SQL 명령의 조회 결과 레코드 한 개당 Student 객체 한 개가 생성되고,

조회 결과 레코드의 컬럼들이 Student 객체의 속성에 채워진다.

이때 컬럼명과 객체의 속성명이 일치하는 것만 채워지고, 일치하지 않는 것은 무시된다.

이렇게 생성된 Student 객체들이 List<Student> 타입의 객체에 채워져 리턴된다.

조회 결과 레코드가 없을 경우에는 목록의 크기 0 인 List<Student> 객체가 리턴된다.

List 객체의 size() 메소드는 목록의 크기를 리턴한다.

(참고: Java에서 List는 클래스가 아니고 interface 이다.

List interface를 구현한 대표적인 클래스가 ArrayList 이다.

<https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/List.html>)

### Java 객체의 속성

Java 객체의 속성은 멤버 변수가 아니다.

get 메소드와 set 메소드가 Java 객체의 속성이다.

예를 들어 Student 클래스에는 getStudentNo 메소드와 setStudentNo 메소드가 있다.

따라서 Student 클래스에는 studentNo 속성이 있다.

studentNo 속성 값을 읽을 때는 getStudentNo 메소드가 호출되고,

studentNo 속성에 값을 대입할 때는, setStudentNo 메소드가 호출된다.

get 메소드와 set 메소드를 전문 용어로 getter, setter 이라고 한다.

### 조회 결과 컬럼명과 속성명 일치

Graphical user interface, text, application, email

Description automatically generated

|  |  |
| --- | --- |
| 조회 결과 컬럼명 | Student 객체의 set 메소드 |
| id | setId |
| studentNo | setStudentNo |
| name | setName |
| departmentId | setDepartmentId |
| phone | setPhone |
| sex | setSex |
| email | setEmail |
| departmentName | setDepartmentName |

|  |
| --- |
| SELECT s.\*, d.name departmentName  FROM student s JOIN department d ON s.departmentId = d.id |

student 테이블의 모든 레코드를 조회하면서,

학생의 소속 학과를 department 테이블에서 찾아서, 학과명도 같이 조회한다.

student 테이블의 departmentId 필드는, 학생이 소속된 학과의 id 값이다.

FROM student s JOIN department d

student 테이블을 조회하면서, department 테이블을 참조한다.

student 테이블의 약칭을 s 로 하고,

department 테이블의 약칭을 d 라고 하겠다.

ON s.departmentId = d.id

student 테이블을 조회하면서 department 테이블을 참조할 때,

student 테이블의 departmentId 필드와, deaprtment 테이블의 id 필드를 비교한다.

즉 학생(studnet)의 소속 학과(department)를 찾는다.

SELECT s.\*, d.name departmentName

student 테이블의 모든 필드와, department 테이블의 name 필드를 출력한다 (조회한다)

department 테이블의 조회 결과 컬럼명을 name 대신 departmentName 이라고 한다.

|  |
| --- |
| SELECT s.\*, d.name departmentName  FROM student s JOIN department d ON s.departmentId = d.id |

위 SQL 에서 departmentName 부분을 생략하면

department 테이블의 name 필드의 조회 결과 컬럼명은 필드명과 같은 name 이다.

student 테이블에도 name 필드가 있기 때문에,

조회 결과에 name 컬럼이 두 개가 된다.

따라서 department 테이블의 조회 결과 컬럼명을 name 대신 departmentName 으로 바꿔야 한다.

Graphical user interface, text

Description automatically generated

## StudentController.java 생성

|  |  |
| --- | --- |
| Package | net.skhu.controller |
| Name | StudentController |

### src/main/java/net/skhu/controller/StudentController.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.StudentMapper;  @Controller  public class StudentController {  @Autowired StudentMapper studentMapper;  @RequestMapping("student/list")  public String list(Model model) {  model.addAttribute("students", studentMapper.findAll());  return "student/list";  }  } |

### @Autowired

@Autowired StudentMapper studentMapper;

StudentMapper 인터페이스를 구현한 Java 클래스를, spring mybatis가 자동으로 구현해 주고,

그 클래스의 객체를 한 개 생성하여, studentMapper 멤버 변수에 자동으로 대입(autowired)해 준다.

그래서 studentMapper 멤버 변수를 선언했을 뿐이고, 어떤 객체도 대입해 주지 않았지만,

mybatis spring이 자동으로 생성해준 객체가 이 멤버 변수에 자동으로 대입되어 있기 때문에,

액션 메소드에서 이 객체를 사용할 수 있다.

model.addAttribute("students", studentMapper.findAll());

studentMapper.findAll() 메소드는 List<Student> 객체를 리턴한다.

여기에는 DB에서 조회한 학생 목록이 들어있다.

이 학생 목록을 students 이름의 model attribute로 뷰에 전달한다.

## student/list.jsp 생성

### src/main/webapp/WEB-INF/views/student/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  45  46 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <style>  div.container { width: 900px; margin: 50px auto; }  thead th { background-color: #eee; }  table{ border-collapse: collapse; width: 100%; }  td, th { padding: 4px; border: 1px solid lightgray; }  td:nth-child(4) { text-align: center; }  </style>  </head>  <body>  <div class="container">  <h1>학생목록</h1>    <table class="table table-bordered table-condensed">  <thead>  <tr>  <th>학번</th>  <th>이름</th>  <th>학과</th>  <th>전화</th>  <th>성별</th>  <th>이메일</th>  </tr>  </thead>  <tbody>  <c:forEach var="student" items="${ students }">  <tr>  <td>${ student.studentNo }</td>  <td>${ student.name }</td>  <td>${ student.departmentName }</td>  <td>${ student.phone }</td>  <td>${ student.sex }</td>  <td>${ student.email }</td>  </tr>  </c:forEach>  </tbody>  </table>  </div>  </body>  </html> |

<c:forEach var="student" items="${ students }">

...

</c:forEach>

students 이름의 model attribute 값은 객체 목록이어야 한다.

이 객체 목록의 객체 각각에 대해서,

<tr>...</tr> 태그들이 반복 출력된다.

이때, 그 각각의 객체는 student 이름의 변수에 대입된다.

<c:forEach> 확장 태그를 사용하려면,

아래의 선언 태그가 파일 선두에 있어야 한다.

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

## 실행

### http://localhost:8088/student/list

Table

Description automatically generated with medium confidence

# 실습

student2 데이터베이스의 department 테이블에 학과 목록이 들어있다.

학과 목록 출력 페이지를 구현하라.

Department DTO 클래스

DepartmentMapper mybatis mapper

DepartmentController 컨트롤러

department/list.jsp 뷰