



03 Mybatis Project

<input checked="" type="checkbox"/> 복습 여부	<input type="checkbox"/>
-------------------------------------------	--------------------------

Project 생성



New Spring Starter Project

Service URL

https://start.spring.io

Name

mybatis

☒ Use default location

Location

C:\Users\SamSung\Desktop\2024\Backend\myba

Browse

Type:

Maven

Packaging:

War

Java Version:

17

Language:

Java

Group

com.example

Artifact

mybatis

Version

0.0.1-SNAPSHOT

Description

Demo project for Spring Boot

Package

net.skhu


Working sets

☐ Add project to working sets

New...

Working sets:

Select...

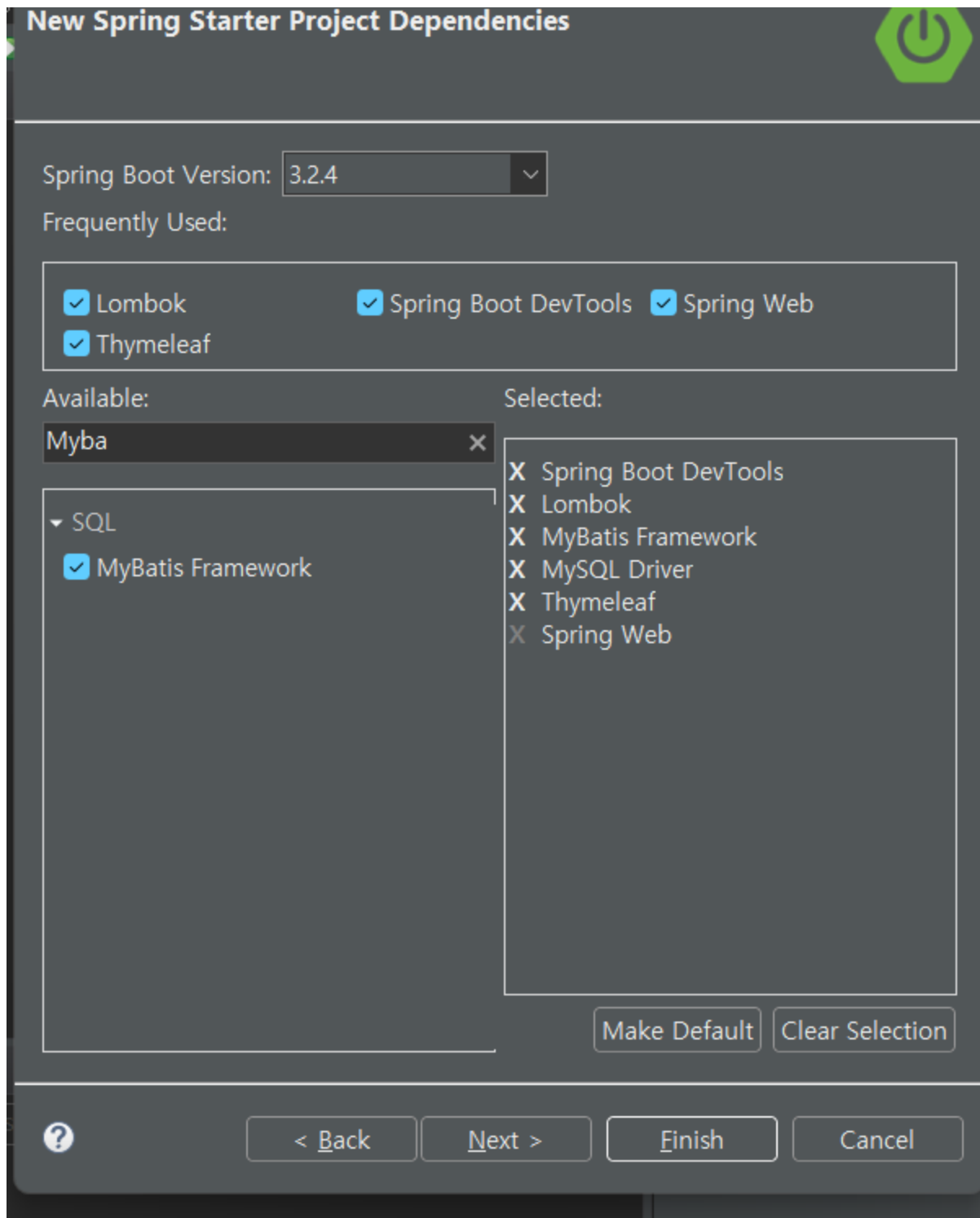


< Back

Next >

Finish

Cancel



Mapper 구현 → CRUD에 관한 SQL 명령들을 구현하기 위한 Mapper interface

- interface임을 주의 → 메소드 본문을 구현하지 않음

- Mapper를 구현해서 훨씬 편리해짐 ? → SqlSession을 등록하지 않아도 됨 등등,,

코드 작성

application.properties 설정 (프로젝트 설정하는 파일)

```
spring.application.name=mybatis
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/student2?useU
spring.datasource.username=user1
spring.datasource.password=skhuA+4.5
server.port=8088
```

DTO 작성

- Student dto 작성

```
package net.skhu.dto;

import lombok.Data;

//lombok이 자동으로 get,set 메소드 구현 ( @annotation → @Data)
@Data
public class Student {
    int id;
    String studentNo;
    String name;

    int departmentId;
    String phone;
    String sex;
    String email;

    String departmentName;
}
```

- Department dto 작성

```
package net.skhu.dto;

import lombok.Data;

@Data
public class Department {
    int id;
    String name;
    String shortName;
    String phone;
}
```

Controller 작성



컨트롤러는 클라이언트로부터 받을 요청을 처리하고, 데이터를 모델에 추가하거나 데이터베이스와의 상호작용을 통해 필요한 작업을 수행한 후, 적절한 뷰를 반환하는 역할을 한다.

- studentController (학생 정보를 관리하는 기능을 담당하는 기본적인 CRUD를 구현한 컨트롤러)

```
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.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
```

```

import net.skhu.dto.Department;
import net.skhu.dto.Student;
import net.skhu.mapper.DepartmentMapper;
import net.skhu.mapper.StudentMapper;

//Spring MVC 컨트롤러임을 나타냄
@Controller

// 이 컨트롤러의 모든 핸들러메소드에 기본적으로 student경로가 적용됨을 나타냄
@RequestMapping("student")
public class StudentController {
    @Autowired // studentMapper interface의 자식 클래스와 메소드들을 주입받음
    StudentMapper studentMapper;
    @Autowired
    DepartmentMapper departmentMapper;

    // 학생 목록을 조회해 students라는 이름으로 모델에 추가한 후, student/list로 이동
    @GetMapping("list")
    public String list(Model model) {
        List<Student> students = studentMapper.findAll();
        model.addAttribute("students", students);
        return "student/list";
    }

    @GetMapping("create")
    public String create(Model model) {
        Student student = new Student();
        List<Department> departments = departmentMapper.findAll();
        model.addAttribute("student", student);
        model.addAttribute("departments", departments);
        return "student/edit";
    }

    //form에서 전송된 학생 정보를 받아 새 학생정보를 DB에 추가한 후, 학생정보를 보여줌
    @PostMapping("create")
    public String create(Model model, Student student) {

```

```

        studentMapper.insert(student);
        return "redirect:list";
    }

    @GetMapping("edit")
    public String edit(Model model, int id) {
        Student student = studentMapper.findOne(id);
        List<Department> departments = departmentMapper.find/
        model.addAttribute("student", student);
        model.addAttribute("departments", departments);
        return "student/edit";
    }

    @PostMapping("edit")
    public String edit(Model model, Student student) {
        studentMapper.update(student);
        return "redirect:list";
    }

    @GetMapping("delete")
    public String delete(Model model, int id) {
        studentMapper.delete(id);
        return "redirect:list";
    }
}

```

Mapper 작성



Mapper는 Mybatis(프로젝트 이름) 라이브러리를 사용하여 DB의 테이블과 상호작용을 위한 인터페이스를 정의

- StudentMapper

- DB의 student 테이블에 대한 조회, 삽입, 수정, 삭제 SQL 명령을 studnetMapper에 구현함

```
package net.skhu.mapper;

import java.util.List;

import org.apache.ibatis.annotations.Delete;
import org.apache.ibatis.annotations.Insert;
import org.apache.ibatis.annotations.Mapper;
import org.apache.ibatis.annotations.Options;
import org.apache.ibatis.annotations.Select;
import org.apache.ibatis.annotations.Update;

import net.skhu.dto.Student;

@Mapper
public interface StudentMapper {
    /* findOne 함수 -> SQL 명령이 조회한 데이터를 Student 객체에 채워줌
    조회 결과 컬럼 이름과 Student 객체의 속성 이름이 일치해야한다. 일치
    @Select("SELECT * FROM student WHERE id=#{id}")
    Student findOne(int id);

    // 리턴 값이 List<Student>임 조회한 레코드를 student 객체에 채워줌
    @Select("""
        SELECT s.*, d.name departmentName
        FROM student s LEFT JOIN department d ON s.departmentId = d.id
    """)
    List<Student> findAll();

    @Insert("""
        INSERT student (studentNo, name, departmentId, pl
        VALUES ({studentNo}, {name}, {departmentId}, #
    """)
    // insert할 테이블의 기본키 필드 이름이 id이고, 이 필드의 값은 새
    @Options(useGeneratedKeys = true, keyProperty = "id")
    void insert(Student student);
```



```

// studentNo = #{studentNo}에서 studentNo의 피라미터 값은 pr
@Update("""
        UPDATE student SET
        studentNo=#{studentNo},
        name=#{name},
        departmentId=#{departmentId},
        phone=#{phone},
        sex=#{sex},
        email=#{email}
        WHERE id = #{id} """)
void update(Student student);

@Delete("DELETE FROM student WHERE id =#{id}")
void delete(int id);
}

```

- DepartmentMapper

```

package net.skhu.mapper;

import java.util.List;

import org.apache.ibatis.annotations.Mapper;
import org.apache.ibatis.annotations.Select;

import net.skhu.dto.Department;

@Mapper
public interface DepartmentMapper {
    @Select("SELECT * FROM department")
    List<Department> findAll();
}

```

View 작성 (html)

- list.html

```

<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="/common.css" />
<style>
a.btn {
    float: right;
    margin-top: j-40px;
}
</style>
<title>Insert title here</title>
</head>
<body>
    <div class="container">
        <h1>학생목록</h1>
        <a href="create" class="btn">학생등록</a>
        <table class="list">
            <thead>
                <tr>
                    <th>id</th>
                    <th>학번</th>
                    <th>이름</th>
                    <th>학과</th>
                    <th>전화</th>
                    <th>성별</th>
                    <th>이메일</th>
                </tr>
            </thead>
            <tbody>
                <tr th:each="st:${students}">
                    <td th:text="${st.id}"></td>
                    <td><a th:text="${st.studentNo}" th:href=
                    <td th:text="${st.name}"></td>

```

```

        <td th:text="${st.departmentName}"></td>
        <td th:text="${st.phone}"></td>
        <td th:text="${st.sex}"></td>
        <td th:text="${st.email}"></td>
    </tr>
</tbody>
</table>
</div>
</body>
</html>

```

- edit.html



Get 요청은 자료 조회 요청에서만 사용해야한다. 삭제 기능은 GET이 아닌, POST 요청이나 DELETE 요청으로 구현해야하는 것이 바람직하다. (실제 서버까지 전달 되지 않을 수 있기 때문이다.)

```

<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="EUC-KR">
<link rel="stylesheet" type="text/css" href="/common.css" />
<title>edit page</title>
<style>
form {
    width: 600px;
    margin: auto;
    padding: 5px 20px;
    box-shadow: 2px 2px 5px gray;
}

td {
    min-width: 100;
    padding: 5px;
}

```

```

}

td:nth-child(1) {
    text-align: right;
}

button {
    margin: 5px 0 20px 20px;
}
</style>
</head>
<body>
    <div class="container">
        // 핑크색은 뷰에 전달된 model attribute 데이터이다. (학생 객체)
        <form method="post" th:object="${student}">
            // student
            <h1 th:text="${student.id}>0 ? '학생수정' : '학생 등록'>
            <table>
                <tr>
                    <td>학번:</td>
                    /* 입력 폼이 출력 될 때, student 객체의 studentNo request parameter의 값은 studentNo이다.
                    */
                    <td><input type="text" th:field="*{studentNo}" />
                </tr>
                <tr>
                    <td>이름:</td>
                    <td><input type="text" th:field="*{name}" />
                </tr>
                <tr>
                    <td>학과:</td>
                    //departments 객체 목록에 있는 Department 객체
                    <td><select th:field="*{departmentId}">
                        <option th:each="dt:${departments}" th:text="${dt.name}"></option>
                    </td>
                </tr>
            </table>
        </form>
    </div>

```

```

        <tr>
            <td>전화:</td>
            <td><input type="text" th:field="*{phone}" /></td>
        </tr>
        <tr>
            <td>성별:</td>
            <td><input type="text" th:field="*{sex}" /></td>
        </tr>
        <tr>
            <td>이메일:</td>
            <td><input type="text" th:field="*{email}" /></td>
        </tr>
    </table>
    <hr />
    <button type="submit" class="btn">저장</button>

    <a th:if="${student.id}>0}" th:href="${'delete?id=${student.id}'}"
        class="btn" onclick="return confirm( '삭제하시겠습니까? ');">삭제</a>
    <a href="list" class="btn">목록으로</a>
</form>
</div>
</body>
</html>

```

CSS 작성

```

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: #ccc;}

input {padding : 4px;}
input {padding: 4px;}
```

실행결과

메인 화면

학생목록

학생등록

id	학번	이름	학과	전화	성별	이메일
1	201532006	최하은	소프트웨어공학과	010-4361-1145	여	choi064@skhu.ac.kr
2	201532007	김진영	정보통신공학과	010-3415-1238	남	kim073@skhu.ac.kr
3	201532008	나철진	정보통신공학과	010-8703-1239	남	na088@skhu.ac.kr
4	201532009	이익수	글로벌IT공학과	010-7875-1251	남	lee097@skhu.ac.kr
5	201532010	이제문	소프트웨어공학과	010-7700-1333	남	lee107@skhu.ac.kr
6	201532011	김영우	소프트웨어공학과	010-2090-1421	남	kim112@skhu.ac.kr
7	201532012	주한요	정보통신공학과	010-4624-1467	남	joo124@skhu.ac.kr
8	201532013	김숙홍	소프트웨어공학과	010-3791-1522	남	kim133@skhu.ac.kr
9	201532014	홍영수	글로벌IT공학과	010-2897-1525	남	hong142@skhu.ac.kr
10	201532015	박원종	소프트웨어공학과	010-7655-1724	남	park157@skhu.ac.kr
11	201432015	변준호	소프트웨어공학과	010-2245-1750	남	byeon152@skhu.ac.kr
12	201532016	고희정	정보통신공학과	010-5691-1943	여	ko165@skhu.ac.kr
13	201432016	신철대	글로벌IT공학과	010-3221-1956	남	sin163@skhu.ac.kr
14	201532017	서윤정	정보통신공학과	010-4310-1965	여	seo174@skhu.ac.kr
15	201432017	오화순	정보통신공학과	010-8527-2048	여	oh178@skhu.ac.kr
16	201532018	박재인	소프트웨어공학과	010-2616-2144	남	park182@skhu.ac.kr
17	201532019	최현복	컴퓨터공학과	010-7312-2171	남	choi197@skhu.ac.kr
18	201432019	임봉영	컴퓨터공학과	010-7475-2192	남	lim197@skhu.ac.kr
19	201532020	장영만	컴퓨터공학과	010-1527-2270	남	jang201@skhu.ac.kr
20	201432020	박주용	글로벌IT공학과	010-1640-2396	남	park201@skhu.ac.kr
21	201532021	황하지	컴퓨터공학과	010-6682-2463	남	hwang216@skhu.ac.kr
22	201532022	유환치	글로벌IT공학과	010-3951-2464	남	yu223@skhu.ac.kr
23	201532023	성희지	소프트웨어공학과	010-4267-2555	남	seong234@skhu.ac.kr

학생 등록

학생 등록

학번:

이름:

학과:

전화:

성별:

이메일:

저장

목록으로

학생 수정

학생수정

학번:

이름:

학과:

전화:

성별:

이메일:

저장

삭제

목록으로