JPA single-valued association

2021-11-13

이승진

**학습목표**

single valued association 구현

**목차**

[1. 배경지식 2](#_Toc87714938)

[1) student2 데이터베이스 2](#_Toc87714939)

[2) Single-Valued Association 구현 2](#_Toc87714940)

[2. jpa2 프로젝트 생성 4](#_Toc87714941)

[1) pom.xml 6](#_Toc87714942)

[2) application.properties 6](#_Toc87714943)

[3) Department 엔터티 클래스 7](#_Toc87714944)

[4) Student 엔터티 클래스 8](#_Toc87714945)

[5) Department 리포지토리 9](#_Toc87714946)

[6) Student 리포지토리 9](#_Toc87714947)

[7) StudentController 컨트롤러 10](#_Toc87714948)

[8) common.css 12](#_Toc87714949)

[9) common.js 12](#_Toc87714950)

[10) studnet/list.jsp 13](#_Toc87714951)

[11) student/edit.jsp 14](#_Toc87714952)

[3. 연습문제 16](#_Toc87714953)

[1) book2 데이터베이스 16](#_Toc87714954)

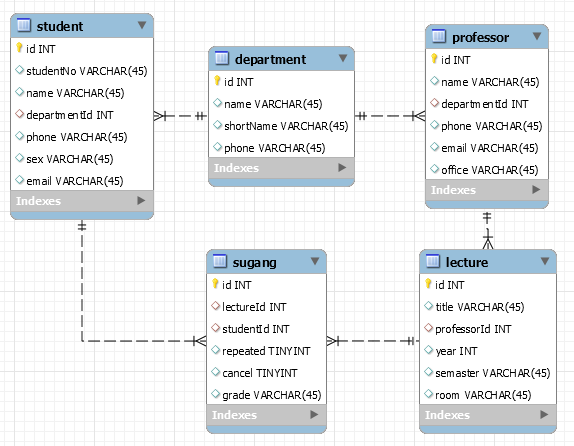
[2) book/list 16](#_Toc87714955)

[3) book/create 17](#_Toc87714956)

[4) book/edit 17](#_Toc87714957)

# 배경지식

## student2 데이터베이스



department 와 student 관계는 1 대 다 관계이다.

student 테이블의 departmentId 필드는 소속 학과를 가르키는 외래키(foreign key) 이다.

## Single-Valued Association 구현

department 와 student 관계의 구현은

외래키(departmentId)가 포함된 student 객체에 department 멤버 변수를 구현한다.

### @ManyToOne 어노테이션

|  |
| --- |
| @Entity  public class Student {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  int id;  String name;  @ManyToOne  @JoinColumn(name = "departmentId")  Department department; |

학생 여러명이 학과 한 개에 소속되어 있다. @ManyToOne

학생 테이블의 외래 키는 departmentId 필드이다. @JoinColumn(name = "departmentId")

학생이 속한 학과는 한 개이다. Department department

departmentId 속성을 만들지 않음에 주목하자.(중요!)<-외래키는 삽입하지 않는다(대신 객체가 들어감)

Student 와 Department 관계는 Many To One 이다.

그래서 Student 클래스의 department 멤버 변수에 @ManyToOne 어노테이션을 붙여야 한다.

이 관계는 departmentId 외래키로 표현된다.

student 테이블과 department 테이블을 조인할 때,

departmentId 외래키로 조인해야 한다.

@ManyToOne

@JoinColumn(name = "departmentId")

Department department;

### 요약

엔터티 클래스들 사이의 1 대 다 관계를 구현할 때,

외래키를 포함하는 엔터티 클래스에 @ManyToOne 어노테이션을 사용하여,

상대측 엔터티 객체 한 개를 구현한다.

그리고 외래키 속성은 생략한다.

예: departmentId 외래키가 student 테이블에 포함되어 있으므로

Student 클래스에 다음과 같은 멤버 변수를 구현한다.

그리고 departmentId 속성은 생략한다.

@ManyToOne

@JoinColumn(name = "departmentId")

Department department;

위와 같이 departmentId 외래키 대신 Department department 멤버 변수를

Student 엔터티 클래스에 정의하면 다음과 같은 일이 자동으로 일어난다.

JPA 엔진이, student 테이블에서 학생 레코드를 조회해서 Student 엔터티 객체를 생성할 때,

Student 객체의 department 속성에 학과 정보도 채워준다.

즉 JPA 엔진이, student 테이블에서 학생 레코드를 조회해서 Student 엔터티 객체를 생성할 때,

그 학생의 departmentId 외래키로 department 테이블도 조회해서,

Department 엔터티 객체를 생성하여,

Student 객체의 department 속성에 Department 객체도 채워준다.

# jpa2 프로젝트 생성

Graphical user interface, text, application, email

Description automatically generated

|  |  |
| --- | --- |
| Name | jpa2 |
| Package | net.skhu |

Graphical user interface, text, application

Description automatically generated

Developer Tools 항목 아래

Spring Boot DevTools

Lombok

SQL 항목 아래

Spring Data JPA

MySQL Driver

Web 항목 아래

Spring Web

## 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/student2?useUnicode=yes&characterEncoding=UTF-8&allowMultiQueries=true&serverTimezone=Asia/Seoul  spring.datasource.username=user1  spring.datasource.password=skhuA+4.5  spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl  logging.level.org.hibernate.SQL=DEBUG  logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE  server.port=8088 |

## Department 엔터티 클래스

### src/main/java/net/skhu/entity/Department.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.entity;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import lombok.Data;  @Data  @Entity  public class Department {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  int id;  String name;  String shortName;  String phone;  } |

## Student 엔터티 클래스

### src/main/java/net/skhu/entity/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 | package net.skhu.entity;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.JoinColumn;  import javax.persistence.ManyToOne;  import lombok.Data;  @Data  @Entity  public class Student {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  int id;  String studentNo;  String name;  String phone;  String sex;  String email;  @ManyToOne  @JoinColumn(name = "departmentId")  Department department;  } |

## Department 리포지토리

### src/main/java/net/skhu/repository/DepartmentRepository.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | package net.skhu.repository;  import org.springframework.data.jpa.repository.JpaRepository;  import net.skhu.entity.Department;  public interface DepartmentRepository extends JpaRepository<Department, Integer> {  } |

## Student 리포지토리

### src/main/java/net/skhu/repository/StudentRepository.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | package net.skhu.repository;  import org.springframework.data.jpa.repository.JpaRepository;  import net.skhu.entity.Student;  public interface StudentRepository extends JpaRepository<Student, Integer> {  } |

## StudentController 컨트롤러

### 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  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 | 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 org.springframework.web.bind.annotation.RequestParam;  import net.skhu.entity.Department;  import net.skhu.entity.Student;  import net.skhu.repository.DepartmentRepository;  import net.skhu.repository.StudentRepository;  @Controller  @RequestMapping("student")  public class StudentController {  @Autowired StudentRepository studentRepository;  @Autowired DepartmentRepository departmentRepository;  @RequestMapping("list")  public String list(Model model) {  List<Student> students = studentRepository.findAll();  model.addAttribute("students", students);  return "student/list";  }  @GetMapping("create")  public String create(Model model) {  Student student = new Student();  List<Department> departments = departmentRepository.findAll();  model.addAttribute("student", student);  model.addAttribute("departments", departments);  return "student/edit";  }  @PostMapping("create")  public String create(Model model, Student student) {  studentRepository.save(student);  return "redirect:list";  }  @GetMapping("edit")  public String edit(Model model, @RequestParam("id") int id) {  Student student = studentRepository.findById(id).get();  List<Department> departments = departmentRepository.findAll();  model.addAttribute("student", student);  model.addAttribute("departments", departments);  return "student/edit";  }  @PostMapping("edit")  public String edit(Model model, Student student) {  studentRepository.save(student);  return "redirect:list";  }  @RequestMapping("delete")  public String delete(Model model, @RequestParam("id") int id) {  studentRepository.deleteById(id);  return "redirect:list";  }  } |

mybatis로 구현했던 StudentController와 별 차이가 없다.

메소드 이름만 다른 정도.

## 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; } |

## common.js

### src/main/resources/static/common.js

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | $(function() {    $("[data-url]").click(function() {  var url = $(this).attr("data-url");  location.href = url;  })  $("[data-confirm-delete]").click(function() {  return confirm("삭제하시겠습니까?");  })  }) |

## studnet/list.jsp

### 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  47  48 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <c:url var="R" value="/" />  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <script src="http://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>  <script src="${R}common.js"></script>  <link rel="stylesheet" type="text/css" href="${R}common.css" />  <style>  a.btn { float: right; margin: -20px 0 5px 0; }  td:nth-child(1), td:nth-child(5) { text-align: center; }  </style>  </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>  <c:forEach var="student" items="${ students }">  <tr data-url="edit?id=${ student.id }">  <td>${ student.id }</td>  <td>${ student.studentNo }</td>  <td>${ student.name }</td>  <td>${ student.department.name }</td>  <td>${ student.phone }</td>  <td>${ student.sex }</td>  <td>${ student.email }</td>  </tr>  </c:forEach>  </tbody>  </table>  </div>  </body>  </html> |

${ student.department.name }

student 객체에 department 속성이 있고, 이 속성의 값은 Department 객체이다.

Department 객체에는 name 속성값이 있다.

## student/edit.jsp

### src/main/webapp/WEB-INF/views/student/edit.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  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67 | <%@ 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://www.springframework.org/tags/form" prefix="form" %>  <c:url var="R" value="/" />  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <script src="http://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>  <script src="${R}common.js"></script>  <link rel="stylesheet" href="${R}common.css">  <style>  form { width: 600px; margin: auto;  box-shadow: 0 0 4px lightgray, 2px 2px 4px gray; overflow: auto; }  div.title { font-size: 20pt; padding: 10px; background-color: #eee; }  table { margin: 20px; }  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">  <form:form method="post" modelAttribute="student">  <div class="title">학생 ${ student.id > 0 ? "수정" : "등록" }</div>  <table>  <tr>  <td>학번:</td>  <td><form:input path="studentNo" /></td>  </tr>  <tr>  <td>이름:</td>  <td><form:input path="name" /></td>  </tr>  <tr>  <td>학과:</td>  <td><form:select path="department.id"  itemValue="id" itemLabel="name" items="${ departments }" /></td>  </tr>  <tr>  <td>전화:</td>  <td><form:input path="phone" /></td>  </tr>  <tr>  <td>성별:</td>  <td>  <form:radiobutton path="sex" value="남" label="남자" />  <form:radiobutton path="sex" value="여" label="여자" />  </td>  </tr>  <tr>  <td>이메일:</td>  <td><form:input path="email" type="email" /></td>  </tr>  </table>  <hr />  <div>  <button type="submit" class="btn">저장</button>  <c:if test="${ student.id > 0 }">  <a href="delete?id=${ student.id }" class="btn" data-confirm-delete>삭제</a>  </c:if>  <a href="list" class="btn">목록으로</a>  </div>  </form:form>  </div>  </body>  </html> |

<form:form method="post" modelAttribute="student">

<form:select path="department.id"

itemValue="id" itemLabel="name" items="${ departments }" />

이 select 태그에는 student.department.id 속성값과 일치하는 학과가 선택된다.

그리고 입력폼이 subit 될 때, 이 select 태그에 의한 request parameter의 이름은 "department.id" 이다.

이 request parameter는 액션 메소드 파라미터 변수인, Student 객체의 department 속성의 id 속성에

채워져 전달된다.

# 과제

## book2 데이터베이스

Diagram

Description automatically generated

### book 테이블

Table

Description automatically generated

### category 테이블

A picture containing application

Description automatically generated

## book/list

A screenshot of a computer

Description automatically generated with medium confidence

## book/create

Graphical user interface, application

Description automatically generated

## book/edit

Graphical user interface, text, application

Description automatically generated