Spring form validation #2

2023-09-10

이승진

**학습목표**

spring form validation 메시지 수정

데이터 오류 검사를 서비스 클래스에 구현하기

**목차**

[1. 배경지식 2](#_Toc145446203)

[1) properties 에디터 인코딩 설정 2](#_Toc145446204)

[2) ValidationMessages.properties 파일 생성 3](#_Toc145446205)

[3) ModelMapper 4](#_Toc145446206)

[4) 학번 중복 검사 5](#_Toc145446207)

[2. mybatis3 프로젝트 7](#_Toc145446208)

[1) 프로젝트 생성 7](#_Toc145446209)

[2) pom.xml 8](#_Toc145446210)

[3) application.properties 9](#_Toc145446211)

[4) ValidationMessages.properties 파일 생성 9](#_Toc145446212)

[5) Department DTO 클래스 9](#_Toc145446213)

[6) DepartmentMapper.java 9](#_Toc145446214)

[7) DepartmentService.java 10](#_Toc145446215)

[8) Student DTO 클래스 10](#_Toc145446216)

[9) StudentEdit 모델 클래스 11](#_Toc145446217)

[10) Student Mapper 구현 12](#_Toc145446218)

[11) StudentService 클래스 13](#_Toc145446219)

[12) Student Controller 구현 14](#_Toc145446220)

[13) 정적 컨텐츠 16](#_Toc145446221)

[14) student/list 뷰 구현 17](#_Toc145446222)

[15) student/edit 뷰 구현 18](#_Toc145446223)

[16) 실행 20](#_Toc145446224)

[3. 연습 문제 21](#_Toc145446225)

# 배경지식

## properties 에디터 인코딩 설정

기본적으로 \*.properties 파일의 한글이 제대로 보이지 않는다.

\*.properties 파일의 기본 인코딩이 UTF-8이 아니기 때문이다.

UTF-8로 변경해 주어야 한글이 제대로 보인다.

Window - Preferences 메뉴 클릭

General 항목 아래에 Content Types 항목

A screenshot of a computer

Description automatically generated

\

## ValidationMessages.properties 파일 생성

src/main/resources/ValidationMessages.properties

|  |
| --- |
| jakarta.validation.constraints.Size.message=크기가 {min} 이상 {max} 이하이어야 합니다.  jakarta.validation.constraints.Email.message=이메일 주소가 바르지 않습니다.  jakarta.validation.constraints.NotEmpty.message=필수 입력항목입니다.  jakarta.validation.constraints.Max.message={value} 이하이어야 합니다.  jakarta.validation.constraints.Min.message={value} 이상이어야 합니다.  jakarta.validation.constraints.Pattern.message={regexp} 패턴과 일치해야 합니다. |

위와 같은 파일을 생성하면, spring form validation 에러 메시지를 변경할 수 있다.

위 파일의 내용 중에서

jakarta.validation.constraints.Size 부분은 @Size 어노테이션 클래스의 패키지 포함 경로명이고

jakarta.validation.constraints.Email 부분은 @Email 어노테이션 클래스의 패키지 포함 경로명이고

jakarta.validation.constraints.NotEmpty 부분은 @NotEmpty 어노테이션 클래스의 패키지 포함 경로명이다.

어노테이션 클래스의 전체 경로명은

StudentEdit.java 파일에서 import 문장에서 확인할 수 있다.

|  |
| --- |
| import jakarta.validation.constraints.Email;  import jakarta.validation.constraints.NotBlank;  import jakarta.validation.constraints.NotEmpty;  import jakarta.validation.constraints.Size;  import jakarta.validation.constraints.Max;  import jakarta.validation.constraints.Min; |

|  |
| --- |
| jakarta.validation.constraints.Size.message=크기가 {min} 이상 {max} 이하이어야 합니다. |

여기서 {min} {max} 부분은

|  |
| --- |
| @Size(min=2, max=30)  String name; |

@Size 어노테이션에서 min 값과 max 값에 해당한다.

|  |
| --- |
| jakarta.validation.constraints.Max.message={value} 이하이어야 합니다. |

어노테이션에 이름 없는 값이 한 개만 주어진 경우에 그 값의 이름은 value 이다.

|  |
| --- |
| @Max(10)  int amount; |

@Max 어노테이션에서 괄호 안의 값의 이름이 value 이다.

## ModelMapper

StudentEdit 모델 객체를 Student DTO 객체로 변환하거나

거꾸로 Student DTO 객체를 StudentEdit 모델 객체로 변환하기 위해

지난 강의의 StudentService 클래스에 다음 같은 메소드들을 구현하였다.

|  |
| --- |
| public Student toDto(StudentEdit studentEdit) {  Student studentDto = new Student();  studentDto.setId(studentEdit.getId());  studentDto.setStudentNo(studentEdit.getStudentNo());  studentDto.setName(studentEdit.getName());  studentDto.setDepartmentId(studentEdit.getDepartmentId());  studentDto.setEmail(studentEdit.getEmail());  studentDto.setPhone(studentEdit.getPhone());  studentDto.setSex(studentEdit.getSex());  return studentDto;  }  public StudentEdit toEditModel(Student studentDto) {  StudentEdit studentEdit = new StudentEdit();  studentEdit.setId(studentDto.getId());  studentEdit.setStudentNo(studentDto.getStudentNo());  studentEdit.setName(studentDto.getName());  studentEdit.setDepartmentId(studentDto.getDepartmentId());  studentEdit.setEmail(studentDto.getEmail());  studentEdit.setPhone(studentDto.getPhone());  studentEdit.setSex(studentDto.getSex());  return studentEdit;  } |

ModelMapper 클래스를 이용하면 매우 간결하게 구현할 수 있다.

|  |
| --- |
| ModelMapper modelMapper = new ModelMapper();  public Student toDto(StudentEdit studentEdit) {  return modelMapper.map(studentEdit, Student.class);  }  public StudentEdit toEditModel(Student studentDto) {  return modelMapper.map(studentDto, StudentEdit.class);  } |

Student 클래스와 StudentEdit 클래스의 속성들은 서로 이름이 같다.

ModelMapper는 이름이 같은 속성들의 값을 자동으로 복사해준다.

return modelMapper.map(studentEdit, Student.class);

studentEdit 객체의 속성값을 복사해서, Student 객체를 생성하여 리턴한다.

return modelMapper.map(studentDto, StudentEdit.class);

studentDto 객체의 속성값을 복사해서, StudentEdit 객체를 생성하여 리턴한다.

## 학번 중복 검사

지난 강의 예제에서 입력된 학번이 중복 되는지 검사하는 작업을 컨트롤러 클래스에 구현했다.

|  |
| --- |
| @PostMapping("create")  public String create(Model model,  @Valid StudentEdit studentEdit, BindingResult bindingResult) {  if (bindingResult.hasErrors()) {  model.addAttribute("departments", departmentService.findAll());  return "student/edit";  }  Student student2 = studentService.findByStudentNo(studentEdit.getStudentNo());  if (student2 != null) {  bindingResult.rejectValue("studentNo", null, "학번이 중복됩니다.");  model.addAttribute("departments", departmentService.findAll());  return "student/edit";  }  studentService.insert(studentEdit);  return "redirect:list";  }  @PostMapping("edit")  public String edit(Model model,  @Valid StudentEdit studentEdit, BindingResult bindingResult) {  if (bindingResult.hasErrors()) {  model.addAttribute("departments", departmentService.findAll());  return "student/edit";  }  Student student2 = studentService.findByStudentNo(studentEdit.getStudentNo());  if (student2 != null && student2.getId() != studentEdit.getId()) {  bindingResult.rejectValue("studentNo", null, "학번이 중복됩니다.");  model.addAttribute("departments", departmentService.findAll());  return "student/edit";  }  studentService.update(studentEdit);  return "redirect:list";  } |

그런데, 컨트롤러에 어떤 작업을 구현하는 것은 바람직하지 않다.

작업은 서비스 클래스에 구현해야 한다.

다음과 같은 메소드를 StudentService 클래스에 구현하자.

|  |
| --- |
| public boolean hasErrors(StudentEdit studentEdit, BindingResult bindingResult) {  // 이 메소드는 studentEdit 모델 객체에 들어있는 데이터에 오류가 있으면 true를 리턴해야 한다.  if (bindingResult.hasErrors()) return true;  // 이미 spring form validation 에러가 있다면 그냥 true를 리턴한다.  Student student2 = findByStudentNo(studentEdit.getStudentNo());  if (student2 != null && student2.getId() != studentEdit.getId()) {  bindingResult.rejectValue("studentNo", null, "학번이 중복됩니다.");  return true; // 학번이 중복된다면 true를 리턴한다.  }  return false; // 여기까지 왔다면 studentEdit 모델 객체의 데이터에 오류가 없으므로,  // false를 리턴한다.  } |

컨트롤러 클래스에서는, 서비스 클래스에 구현된 hasErrors 메소드를 호출하면 되므로

다음과 같이 구현이 간결해진다.

|  |
| --- |
| @PostMapping("create")  public String create(Model model,  @Valid StudentEdit studentEdit, BindingResult bindingResult) {  if (studentService.hasErrors(studentEdit, bindingResult)) {  model.addAttribute("departments", departmentService.findAll());  return "student/edit";  }  studentService.insert(studentEdit);  return "redirect:list";  }  @PostMapping("edit")  public String edit(Model model,  @Valid StudentEdit studentEdit, BindingResult bindingResult) {  if (studentService.hasErrors(studentEdit, bindingResult)) {  model.addAttribute("departments", departmentService.findAll());  return "student/edit";  }  studentService.update(studentEdit);  return "redirect:list";  } |

# mybatis3 프로젝트

## 프로젝트 생성

A screenshot of a computer

Description automatically generated

|  |  |
| --- | --- |
| Name | mybatis3 |
| Java Version | 17 |
| Package | net.skhu |

A screenshot of a computer

Description automatically generated

I/O 그룹 아래에서 Validation 항목을 체크한다.

## pom.xml

|  |
| --- |
| <dependency>  <groupId>org.modelmapper</groupId>  <artifactId>modelmapper</artifactId>  <version>3.1.1</version>  </dependency> |

ModelMapper maven dependency를 추가하자.

## application.properties

src/main/resources/application.properties

|  |
| --- |
| 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 |

## ValidationMessages.properties 파일 생성

src/main/resources/ValidationMessages.properties

|  |
| --- |
| jakarta.validation.constraints.Size.message=크기가 {min} 이상 {max} 이하이어야 합니다.  jakarta.validation.constraints.Email.message=이메일 주소가 바르지 않습니다.  jakarta.validation.constraints.NotEmpty.message=필수 입력항목입니다.  jakarta.validation.constraints.Max.message={value} 이하이어야 합니다.  jakarta.validation.constraints.Min.message={value} 이상이어야 합니다.  jakarta.validation.constraints.Pattern.message={regexp} 패턴과 일치해야 합니다. |

## Department DTO 클래스

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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | package net.skhu.dto;  import lombok.Data;  @Data  public class Department {  int id;  String name;  String shortName;  String phone;  } |

## DepartmentMapper.java

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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | 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();  } |

## DepartmentService.java

src/main/java/net/skhu/service/DepartmentService.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | package net.skhu.service;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import net.skhu.dto.Department;  import net.skhu.mapper.DepartmentMapper;  @Service  public class DepartmentService {  @Autowired  public DepartmentMapper departmentMapper;  public List<Department> findAll() {  return departmentMapper.findAll();  }  } |

## Student DTO 클래스

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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | package net.skhu.dto;  import lombok.Data;  @Data  public class Student {  int id;  String studentNo;  String name;  int departmentId;  String phone;  String sex;  String email;  String departmentName;  } |

## StudentEdit 모델 클래스

src/main/java/net/skhu/model/StudentEdit.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 | package net.skhu.model;  import jakarta.validation.constraints.Email;  import jakarta.validation.constraints.NotBlank;  import jakarta.validation.constraints.NotEmpty;  import jakarta.validation.constraints.Pattern;  import jakarta.validation.constraints.Size;  import lombok.Data;  @Data  public class StudentEdit {  int id;  @NotEmpty @NotBlank  @Size(min=8, max=12)  String studentNo;  @NotEmpty @NotBlank  @Size(min=2, max=20)  String name;  @NotEmpty @NotBlank  @Pattern(regexp="010-[0-9]{3,4}-[0-9]{4}")  String phone;  @NotEmpty @Email  String email;  @NotEmpty @NotBlank  @Pattern(regexp="남|여")  String sex;  int departmentId;  } |

## Student Mapper 구현

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  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 | 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 {  @Select("SELECT \* FROM student WHERE id = #{id}")  Student findOne(int id);  @Select("SELECT \* FROM student WHERE studentNo = #{studentNo}")  Student findByStudentNo(String studentNo);  @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, phone, sex, email)  VALUES (#{studentNo}, #{name}, #{departmentId}, #{phone}, #{sex}, #{email}) """)  @Options(useGeneratedKeys=true, keyProperty="id")  void insert(Student student);  @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);  } |

mapper와 서비스 클래스 사이에는 DTO 객체만 사용해야 한다.

## StudentService 클래스

src/main/java/net/skhu/service/StudentService.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 | package net.skhu.service;  import java.util.List;  import org.modelmapper.ModelMapper;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import net.skhu.dto.Student;  import net.skhu.mapper.StudentMapper;  import net.skhu.model.StudentEdit;  @Service  public class StudentService {  @Autowired StudentMapper studentMapper;  ModelMapper modelMapper = new ModelMapper();  public StudentEdit findOne(int id) {  Student studentDto = studentMapper.findOne(id);  return toEditModel(studentDto);  }  public Student findByStudentNo(String studentNo) {  return studentMapper.findByStudentNo(studentNo);  }  public List<Student> findAll() {  return studentMapper.findAll();  }  public void insert(StudentEdit studentEdit) {  Student student = toDto(studentEdit);  studentMapper.insert(student);  }  public void update(StudentEdit studentEdit) {  Student student = toDto(studentEdit);  studentMapper.update(student);  }  public void delete(int id) {  studentMapper.delete(id);  }  public Student toDto(StudentEdit studentEdit) {  return modelMapper.map(studentEdit, Student.class);  }  public StudentEdit toEditModel(Student studentDto) {  return modelMapper.map(studentDto, StudentEdit.class);  }  public boolean hasErrors(StudentEdit studentEdit, BindingResult bindingResult) {  if (bindingResult.hasErrors()) return true;  Student student2 = findByStudentNo(studentEdit.getStudentNo());  if (student2 != null && student2.getId() != studentEdit.getId()) {  bindingResult.rejectValue("studentNo", null, "학번이 중복됩니다.");  return true;  }  return false;  }  } |

## Student Controller 구현

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  6  67  68  69  70  71  72  73  74  75  76  77 | 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.validation.BindingResult;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PostMapping;  import org.springframework.web.bind.annotation.RequestMapping;  import jakarta.validation.Valid;  import net.skhu.dto.Department;  import net.skhu.dto.Student;  import net.skhu.model.StudentEdit;  import net.skhu.service.DepartmentService;  import net.skhu.service.StudentService;  @Controller  @RequestMapping("student")  public class StudentController {  @Autowired StudentService studentService;  @Autowired DepartmentService departmentService;  @GetMapping("list")  public String list(Model model) {  List<Student> students = studentService.findAll();  model.addAttribute("students", students);  return "student/list";  }  @GetMapping("create")  public String create(Model model) {  StudentEdit studentEdit = new StudentEdit();  List<Department> departments = departmentService.findAll();  model.addAttribute("studentEdit", studentEdit);  model.addAttribute("departments", departments);  return "student/edit";  }  @PostMapping("create")  public String create(Model model,  @Valid StudentEdit studentEdit, BindingResult bindingResult) {  if (studentService.hasErrors(studentEdit, bindingResult)) {  model.addAttribute("departments", departmentService.findAll());  return "student/edit";  }  studentService.insert(studentEdit);  return "redirect:list";  }  @GetMapping("edit")  public String edit(Model model, int id) {  StudentEdit studentEdit = studentService.findOne(id);  List<Department> departments = departmentService.findAll();  model.addAttribute("studentEdit", studentEdit);  model.addAttribute("departments", departments);  return "student/edit";  }  @PostMapping("edit")  public String edit(Model model,  @Valid StudentEdit studentEdit, BindingResult bindingResult) {  if (studentService.hasErrors(studentEdit, bindingResult)) {  model.addAttribute("departments", departmentService.findAll());  return "student/edit";  }  studentService.update(studentEdit);  return "redirect:list";  }  @GetMapping("delete")  public String delete(Model model, int id) {  studentService.delete(id);  return "redirect:list";  }  } |

## 정적 컨텐츠

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 | 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; }  div.error { color: red; } |

## student/list 뷰 구현

src/main/resources/templates/student/list.html

|  |  |
| --- | --- |
| 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 | <!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: -40px; }  </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>  <tr th:each="st : ${ students }">  <td th:text="${ st.id }"></td>  <td><a th:text="${ st.studentNo }" th:href="${ 'edit?id=' + st.id }"></a></td>  <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> |

## student/edit 뷰 구현

src/main/resources/templates/student/edit.html

|  |  |
| --- | --- |
| 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 | <!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>  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; vertical-align: top; }  button { margin: 5px 0 20px 20px; }  </style>  </head>  <body>  <div class="container">  <form method="post" th:object="${studentEdit}">  <h1 th:text="${studentEdit.id > 0 ? '학생 수정' : '학생 등록'}"></h1>  <table>  <tr>  <td>학번:</td>  <td><input type="text" th:field="\*{studentNo}" />  <div class="error" th:errors="\*{studentNo}"></div>  </td>  </tr>  <tr>  <td>이름:</td>  <td><input type="text" th:field="\*{name}" />  <div class="error" th:errors="\*{name}"></div>  </td>  </tr>  <tr>  <td>학과:</td>  <td><select th:field="\*{departmentId}">  <option th:each="dt : ${ departments }"  th:value="${ dt.id }" th:text="${ dt.name }">  </option>  </select>  <div class="error" th:errors="\*{departmentId}"></div>  </td>  </tr>  <tr>  <td>전화:</td>  <td><input type="text" th:field="\*{phone}" />  <div class="error" th:errors="\*{phone}"></div>  </td>  </tr>  <tr>  <td>성별:</td>  <td><input type="text" th:field="\*{sex}" />  <div class="error" th:errors="\*{sex}"></div>  </td>  </tr>  <tr>  <td>이메일:</td>  <td><input type="text" th:field="\*{email}" />  <div class="error" th:errors="\*{email}"></div>  </td>  </tr>  </table>  <hr />  <div>  <button type="submit" class="btn">저장</button>  <a th:if="${ studentEdit.id > 0 }" th:href="${ 'delete?id=' + studentEdit.id }"  class="btn" onclick="return confirm('삭제하시겠습니까?')">삭제</a>  <a href="list" class="btn">목록으로</a>  </div>  </form>  </div>  </body>  </html> |

## 실행

A screenshot of a computer

Description automatically generated

아무것도 입력하지 않고 저장 클릭하면

# 연습 문제

professor 테이블에 대해서서

spring form validation을 구현하라