Spring Form Validation 2

2021-11-07

이승진

**학습목표**

validation message 수정하기

로직 에러 처리 구현하기

서비스 클래스 구현하기

**목차**

[1. validation message 수정 2](#_Toc87254642)

[1) 개요 2](#_Toc87254643)

[2) Properties Editor 설치 3](#_Toc87254644)

[3) ValidationMessages.properties 생성 4](#_Toc87254645)

[4) ValidationMessage.properties 5](#_Toc87254646)

[5) 실행 6](#_Toc87254647)

[2. 개별 항목에 대한 에러 메시지 등록 7](#_Toc87254648)

[1) 목표 7](#_Toc87254649)

[2) message 애트리뷰트 7](#_Toc87254650)

[3) UserRegistration.java 수정 8](#_Toc87254651)

[4) 실행 9](#_Toc87254652)

[3. 로직 에러 처리하기 10](#_Toc87254653)

[1) 목표 10](#_Toc87254654)

[2) UserService.java 11](#_Toc87254655)

[3) UserController.java 12](#_Toc87254656)

[4) 실행 13](#_Toc87254657)

# validation message 수정

## 개요

Graphical user interface

Description automatically generated

위 화면에 자동으로 출력된 에러 메시지 문구를 수정하자.

각 spring validation annotation 에 대한 에러 메시지를

ValidationMessages.properties 파일에 등록해 주면 된다.

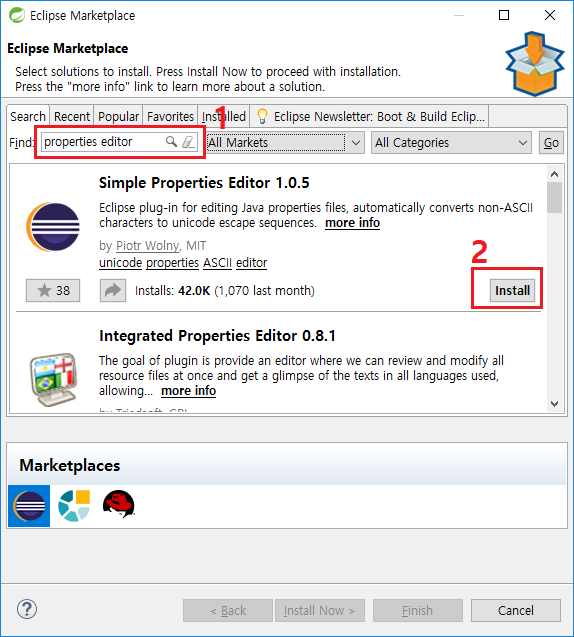
지난 강의에서 만든 form2 프로젝트를 수정하자.

## Properties Editor 설치

ValidationMessages.properties 파일에 한글을 입력하려면,

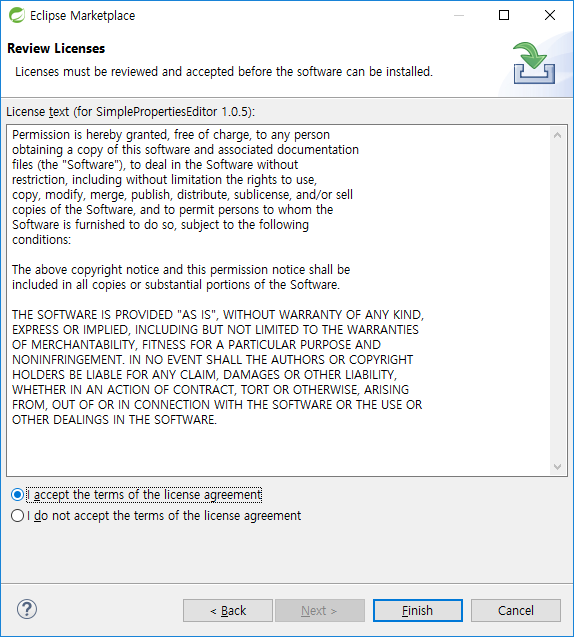
PropertiesEditor가 필요하다.

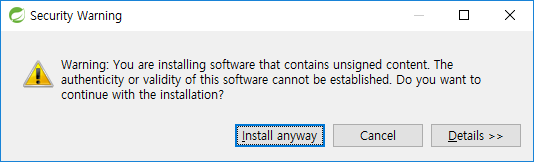
메뉴: Help - Eclipse Market Place

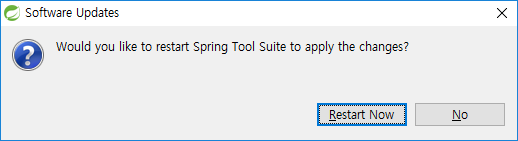


1) 검색 창에 "properties editor" 를 입력하고 엔터키를 누른다.

2) 검색 결과에서 Properties Editor 항목을 찾아서 Install 버튼을 클릭한다.





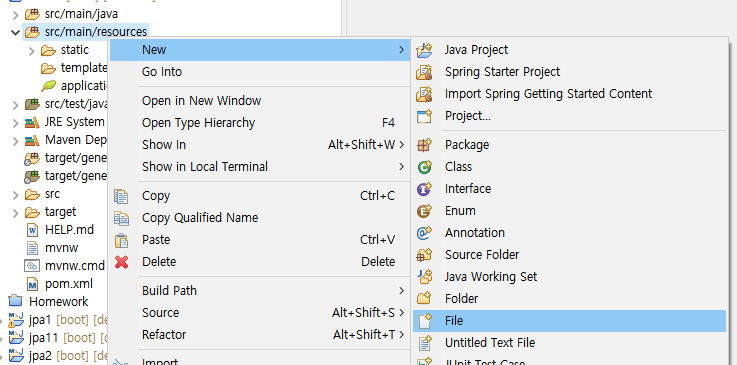


## ValidationMessages.properties 생성

src/main/resources/ValidationMessages.properties 파일을 생성하자.

1) src/main/resources 폴더를 마우스 오른쪽 버튼을 클릭하고

2) New - File 메뉴 클릭



Graphical user interface, text, application

Description automatically generated

## ValidationMessage.properties

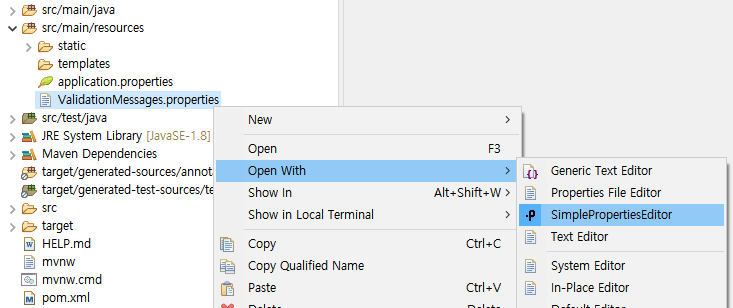
만약 ValidationMessages.properties 파일을 수정할 때 한글이 깨진다면,

그 이유는 이클립스 기본 편집기에서 파일을 열었기 때문이다.

앞에서 설치한 PropertiesEditor 편집기에서 ValidationMessages.properties 파일을 열면

한글이 깨지지 않는다.

아래의 방법으로 ValidationMessage.properties 파일을 열어서 내용을 입력하자.



위 화면의 Open With - SimplePropertiesEditor 메뉴를 클릭하여 ValidationMessages.properties 파일을 열면 된다.

### src/main/resources/ValidationMessages.properties

|  |
| --- |
| javax.validation.constraints.Size.message=크기가 {min} 이상 {max} 이하이어야 합니다.  javax.validation.constraints.Email.message=이메일 주소가 바르지 않습니다.  javax.validation.constraints.NotEmpty.message=필수 입력항목입니다. |

위 파일의 내용 중에서

javax.validation.constraints.Size 부분은 @Size 어노테이션 클래스의 경로명이고

javax.validation.constraints.Email 부분은 @Email 어노테이션 클래스의 경로명이고

javax.validation.constraints.NotEmpty 부분은 @NotEmpty 어노테이션 클래스의 경로명이다.

어노테이션 클래스의 전체 경로명은

UserRegistrationModel.java 파일에서 import 문장에서 확인할 수 있다.

|  |
| --- |
| import javax.validation.constraints.Size;  import javax.validation.constraints.Email;  import javax.validation.constraints.NotEmpty; |

위 파일의 내용 중에서 {min} {max} 부분은

@Size(min=3, max=12) 어노테이션에서 min 값과 max 값에 해당한다.

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

예:

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

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

위와 같은 방법으로 다른 어노테이션 메시지도 등록할 수 있다.

## 실행

Graphical user interface, text

Description automatically generated

에러 메시지가 변경되었다.

# 개별 항목에 대한 에러 메시지 등록

## 목표

@NotEmpty 어노테이션에 위배되는 모든 항목에 "필수 입력항목입니다." 에러 메시지가 동일하게 출력된다.

항목에 따라서 다른 에러 메시지가 표시되어야 하는 경우에는

@NotEmpty 어노테이션에 message 애트리뷰트 값으로 에러 메시지를 등록하면 된다.

## message 애트리뷰트

어떤 항목의 @NotEmpty 에러 메시지를 다른 것으로 바꾸려면,

이 어노테이션에 message 애트리뷰트 값으로 에러 메시지를 등록하면 된다.

예:

|  |
| --- |
| @NotEmpty  @Size(min=6, max=12)  @Min(1) |

위 어노테이션들에 message 애트리뷰트 값을 추가하면 다음과 같다.

|  |
| --- |
| @NotEmpty(message="학번을 입력하세요")  @Size(min=6, max=12, message="학번은 6 자리 이상 12 자리 이하이어야 합니다.")  @Min(value=1, message="양의 정수를 입력하세요") |

@Min(1)

위와 같이 값이 한 개인 어노테이션에서 그 값 한 개인 애트리뷰트 이름은 value 이다.

애트리뷰트 값이 한 개 뿐일 때, value 이름을 생략할 수 있기 때문에

@Min(1) 도 가능하고 @Min(value=1) 도 가능하다.

그런데

값이 여러 개일 때는 애트리뷰트 이름을 생략할 수 없기 때문에

@Min(value=1, message="양의 정수를 입력하세요") 이렇게 적어야 한다.

예를 들어 @RequestMapping("list") 도 가능하고 @RequestMapping(value="list") 도 가능하다.

그런데 @RequestMapping(value="save", method=RequestMethod.POST) 의 경우에

애트리뷰트 이름을 생략할 수 없다.

## UserRegister.java 수정

### src/main/java/net/skhu/model/UserRegister.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 | package net.skhu.model;  import javax.validation.constraints.Email;  import javax.validation.constraints.NotEmpty;  import javax.validation.constraints.Size;  import lombok.Data;  @Data  public class UserRegister {  @NotEmpty(message="사용자 아이디를 입력하세요")  @Size(min=3, max=12)  String userid;  @NotEmpty(message="비밀번호를 입력하세요")  @Size(min=6, max=12, message="비밀번호는 6 자리 이상 12 자리 이하이어야 합니다.")  String passwd1;  @NotEmpty(message="비밀번호를 한번 더 입력하세요")  String passwd2;  @NotEmpty(message="이름을 입력하세요")  @Size(min=2, max=30)  String name;  @NotEmpty(message="이메일 주소를 입력하세요")  @Email(message="이메일 주소가 올바르지 않습니다")  String email;  Integer departmentId;  } |

## 실행

Graphical user interface, text

Description automatically generated

# 로직 에러 처리하기

## 목표

validation annotation 에 의해서 자동으로 검사되기 힘든 에러들도 있다.

예를 들어 사용자 아이디 중복을 검사하려면, DB 의 User 테이블을 조회해야 한다.

입력된 두 비밀번호가 일치하는 지 검사하려면, 두 멤버 변수를 비교해야 한다.

이런 에러 검사는 validation annotation으로 구현하기 힘들고,

따로 직접 구현해야 한다.

### bindingResult 객체에 에러 메시지 등록하기

bindingResult.rejectValue("passwd2", null, "비밀번호가 일치하지 않습니다.");

위와 같이 rejectValue 메소드를 호출하여 에러 메시지를 등록한다.

첫째 파라미터: 에러가 발생한 속성의 이름이다.

셋째 파라미터: 에러 메시지이다.

bindingResult 에 위 에러 메시지가 등록되면, 아래 태그에 그 에러 메시지가 출력된다.

<form:errors path="passwd2" class="error" />

## UserService.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 | package net.skhu.service;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import org.springframework.validation.BindingResult;  import net.skhu.dto.User;  import net.skhu.mapper.UserMapper;  import net.skhu.model.UserRegister;  @Service  public class UserService {  @Autowired  public UserMapper userMapper;  public List<User> findAll() {  return userMapper.findAll();  }  public User findByUserid(String userid) {  return userMapper.findByUserid(userid);  }  public boolean hasErrors(UserRegister userRegister, BindingResult bindingResult) {  if (bindingResult.hasErrors())  return true;  if (userRegister.getPasswd1().equals(userRegister.getPasswd2()) == false) {  bindingResult.rejectValue("passwd2", null, "비밀번호가 일치하지 않습니다.");  return true;  }  User user = userMapper.findByUserid(userRegister.getUserid());  if (user != null) {  bindingResult.rejectValue("userid", null, "사용자 아이디가 중복됩니다.");  return true;  }  return false;  }  public User createDto(UserRegister userRegister) {  User user = new User();  user.setUserid(userRegister.getUserid());  user.setPassword(userRegister.getPasswd1());  user.setName(userRegister.getName());  user.setEmail(userRegister.getEmail());  user.setEnabled(true);  user.setUserType("학생");  if (userRegister.getDepartmentId() != 0)  user.setDepartmentId(userRegister.getDepartmentId());  return user;  }  public void insert(UserRegister userRegister) {  User user = createDto(userRegister);  userMapper.insert(user);  }  } |

## UserController.java

**src/main/java/net/skhu/controller/UserController.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 | package net.skhu.controller;  import javax.validation.Valid;  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 net.skhu.model.UserRegister;  import net.skhu.service.DepartmentService;  import net.skhu.service.UserService;  @Controller  @RequestMapping("user")  public class UserController {  @Autowired UserService userService;  @Autowired DepartmentService departmentService;  @GetMapping("list")  public String list(Model model) {  model.addAttribute("users", userService.findAll());  return "user/list";  }  @GetMapping("register")  public String register(Model model) {  model.addAttribute("userRegister", new UserRegister());  model.addAttribute("departments", departmentService.findAll());  return "user/register";  }  @PostMapping("register")  public String register(Model model,  @Valid UserRegister userRegister, BindingResult bindingResult)  {  if (userService.hasErrors(userRegister, bindingResult)) {  model.addAttribute("departments", departmentService.findAll());  return "user/register";  }  userService.insert(userRegister);  return "redirect:list";  }  } |

## 실행

Graphical user interface, text

Description automatically generated

