spring form1

2021-10-31

이승진

**학습목표**

모델 객체

select 태그 구현

radio 버튼 구현

checkbox 구현

**목차**

[1. form1 프로젝트 2](#_Toc57815556)

[1) 프로젝트 생성 2](#_Toc57815557)

[2) pom.xml 3](#_Toc57815558)

[3) application.properties 4](#_Toc57815559)

[4) Department.java 4](#_Toc57815560)

[5) DepartmentRepository.java 4](#_Toc57815561)

[2. select 태그 구현 5](#_Toc57815562)

[1) 모델 객체 5](#_Toc57815563)

[2) 데이터 객체들 5](#_Toc57815564)

[3) Form1.java 6](#_Toc57815565)

[4) Form1Controller.java 7](#_Toc57815566)

[5) form1/edit1.jsp 8](#_Toc57815567)

[3. radio 버튼 구현 11](#_Toc57815568)

[1) Form1Controller.java 11](#_Toc57815569)

[2) form1/edit2.jsp 12](#_Toc57815570)

[4. checkbox 구현 15](#_Toc57815571)

[1) Form1Controller.java 15](#_Toc57815572)

[2) form1/edit3.jsp 16](#_Toc57815573)

# form1 프로젝트

## 프로젝트 생성

Graphical user interface, text, application, email

Description automatically generated

Name: form1

Graphical user interface, text, application

Description automatically generated

## 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  server.port=8088 |

## Department.java

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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | 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();  } |

## 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  21 | 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; } |

# select 태그 구현

select 태그의 여러가지 구현 방법을 살펴보자.

## 모델 객체

Spring MVC 에서 M은 Model, V는 View, C는 Controller 이다.

모델은 컨트롤러와 뷰 사이에 전달되는 데이터 객체이다.

다음과 같은 데이터들이 모델 객체이다.

(1) 컨트롤러 액션 메소드에서 뷰에게 전달하는 데이터

model.addAttribute("student", studentMapper.findById(id));

(2) 뷰의 실행 결과 출력이 웹브라우저 창에 전달되어 그려질 때,

입력 폼에 채워져 있는 데이터

<form:form modelAttribute="student">

<form:input path="studentNo" /> 이 확장 태그는

==<input type=”text” name=”studentNo” value={$student.studentNo}>와 출력 형식이 동일하다

<form:input path="name" />

<button type="submit">저장</button>

</form>

<form:\*\*\*\*>form:prefix로 시작되는 것들은 spring form 확장 태그이다

뷰에는 평범한 html태그와 확장 태그가 들어있다

평범한 html태그는 그냥 그대로 웹브라우저에 출력된다

확장 태그는(java로 구현됨) 서버에서 실행되고 실행 결과 출력이 웹브라우저에 출력된다

확장태그는 평범한 html태그를 출력한다

확장 태그를 사용하는 이유는 html태그 출력을 간결하게 구현하기 위해서.

(3) 웹브라우저 입력 폼의 내부 submit 버튼이 클릭되어,

입력 폼에 입력된 데이터가 서버의 액션 메소드의 파라미터로 전달될 때,

그렇게 전달되는 데이터

@PostMapping("edit")

String edit(Model model, Student student) {

### DTO, VO

mybatis 예제에서 우리는 DTO 클래스를 사용했다.

DB에서 조회한 데이터를 채워서 전달하는 용도로 사용되는 객체를

DTO (Data Transfer Object)라고 부르는 사람들도 있고

VO (Value Object)라고 부르는 사람들도 있지만,

모델 객체라고 부르는 사람은 거의 없다.

내 생각에는 VO 보다 DTO가 더 적절한 이름이다. (VO는 옛날 이름)

DB에서 조회한 데이터를 전달하기 위해 구현한 DTO 클래스를,

컨트롤러와 뷰 사이에 데이터를 전달하는 용도로 사용하기도 한다.

mybatis 예제에서 그렇게 사용했다.

그런데, DB에서 조회한 데이터를 전달하는 용도로는 사용하지 않고,

뷰와 컨트롤러 사이에 데이터를 전달하는 용도로만 사용하는 객체는

모델 객체라고 부르는 것이 적절하다.

그래서 아래의 Form1 클래스의 package가 dto가 아니고 model 이다.

## Form1.java

### src/main/java/net/skhu/model/Form1.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | package net.skhu.model;  import lombok.Data;  @Data  public class Form1 {  int departmentId1;  int departmentId2;  int departmentId3;  int[] departmentId4;  int[] departmentId5;  boolean enabled;  } |

select 태그의 여러가지 구현 예제에서 뷰와 컨트롤러 사이에 전달될 모델 데이터 객체이다.

### Model

(1) 컨트롤러 액션 메소드에서 뷰에게 전달하는 데이터

(2) 뷰의 실행 결과 출력이 웹브라우저 창에 전달되어 그려질 때,

입력 폼에 채워져 있는 데이터

(3) 웹브라우저 입력 폼의 내부 submit 버튼이 클릭되어,

입력 폼에 입력된 데이터가 서버의 액션 메소드의 파라미터로 전달될 때,

그렇게 전달되는 데이터

뷰에서 아래로 데이터가 왔다갔다함(입력 폼에 가까움)

### DTO

DB에서 조회한 데이터를 채워서 전달하기 위한 객체

(뷰에서 위로 데이터가 왔다갔다함 (DB에 가까움))

## Form1Controller.java

### src/main/java/net/skhu/controller/Form1Controller.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.controller;  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.ModelAttribute;  import org.springframework.web.bind.annotation.PostMapping;  import net.skhu.mapper.DepartmentMapper;  import net.skhu.model.Form1;  @Controller  public class Form1Controller {  @Autowired  DepartmentMapper departmentMapper;  @ModelAttribute  public void addSomething(Model model) {  model.addAttribute("departments", departmentMapper.findAll());  }  @GetMapping("form1/edit1")  public String edit1(Model model) {  model.addAttribute("form1", new Form1());  return "form1/edit1";  }  @PostMapping("form1/edit1")  public String edit1(Model model, Form1 form1) {  return "form1/edit1";  }  } |

@ModelAttribute

public void addSomething(Model model) {

model.addAttribute("departments", departmentRepository.findAll());

}

컨트롤러의 모든 액션 메소드에 똑같은 model.addAttribute(....) 문장을 실행해야 한다면,

그 문장들을 @ModelAttribute 어노테이션이 붙은 메소드에 모아서 구현할 수 있다.

즉 @ModelAttribute 붙은 메소드의 본문의 model.addAttribute(...) 문장들은

이 컨트롤러의 모든 액션 메소드에 적용된다.

즉 다음과 같은 효과가 있다.

@GetMapping("form1/edit1")

public String edit1(Model model) {

model.addAttribute("departments", departmentRepository.findAll());

model.addAttribute("form1", new Form1());

return "form1/edit1";

}

@PostMapping("form1/edit1")

public String edit1(Model model, Form1 form1) {

model.addAttribute("departments", departmentRepository.findAll());

return "form1/edit1";

}

## form1/edit1.jsp

### src/main/webapp/WEB-INF/views/form1/edit1.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 | <%@ 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">  <link rel="stylesheet" type="text/css" href="${R}common.css">  <style>  form { width: 700px; margin: auto;  box-shadow: 0 0 4px lightgray, 2px 2px 4px gray; overflow: auto; }  form div { margin: 20px; }  select { padding: 5px; width: 200px; }  button { margin: 10px; padding: 0.5em 2em; }  ul { margin-top: 30px; }  </style>  </head>  <body>  <div class="container">  <form:form method="post" modelAttribute="form1">  <div>  <form:select path="departmentId1" itemValue="id" itemLabel="name" items="${ departments }" />  </div>  <div>  <form:select path="departmentId2">  <form:option value="0" label="학과를 선택하세요" />  <form:options itemValue="id" itemLabel="name" items="${ departments }" />  </form:select>  </div>  <div>  <form:select path="departmentId3">  <form:option value="0" label="학과를 선택하세요" />  <form:option value="1" label="소프트웨어공학과" />  <form:option value="2" label="컴퓨터공학과" />  <form:option value="3" label="정보통신공학과" />  <form:option value="4" label="글로컬IT공학과" />  </form:select>  </div>  <hr />  <button type="submit">저장</button>  </form:form>    <ul>  <li>departmentId1 : ${ form1.departmentId1 }</li>  <li>departmentId2 : ${ form1.departmentId2 }</li>  <li>departmentId3 : ${ form1.departmentId3 }</li>  </ul>  </div>  </body>  </html> |
|  |  |

<form:select path="departmentId1" itemValue="id" itemLabel="name" items="${ departments }" />

만약 form1 모델 객체의 departmentId1 속성 값이 4 이라면,

위 확장 태그의 실행 결과 출력은 다음과 같다.

<select id="departmentId1" name="departmentId1">

<option value="1">소프트웨어공학과</option>

<option value="2">컴퓨터공학과</option>

<option value="3">정보통신공학과</option>

<option value="4" selected>글로컬IT공학과</option>

</select>

<form:select path="departmentId2">

<form:option value="0" label="학과를 선택하세요" />

<form:options itemValue="id" itemLabel="name" items="${ departments }" />

</form:select>

만약 form1 모델 객체의 departmentId2 속성 값이 1 이라면,

위 확장 태그의 실행 결과 출력은 다음과 같다.

<select id="departmentId2" name="departmentId2">

<option value="0">학과를 선택하세요</option>

<option value="1" selected>소프트웨어공학과</option>

<option value="2">컴퓨터공학과</option>

<option value="3">정보통신공학과</option>

<option value="4">글로컬IT공학과</option>

</select>

<form:select path="departmentId3">

<form:option value="0" label="학과를 선택하세요" />

<form:option value="1" label="소프트웨어공학과" />

<form:option value="2" label="컴퓨터공학과" />

<form:option value="3" label="정보통신공학과" />

<form:option value="4" label="글로컬IT공학과" />

</form:select>

만약 form1 모델 객체의 departmentId3 속성 값이 1 이라면,

위 확장 태그의 실행 결과 출력은 다음과 같다.

<select id="departmentId3" name="departmentId3">

<option value="0">학과를 선택하세요</option>

<option value="1" selected>소프트웨어공학과</option>

<option value="2">컴퓨터공학과</option>

<option value="3">정보통신공학과</option>

<option value="4">글로컬IT공학과</option>

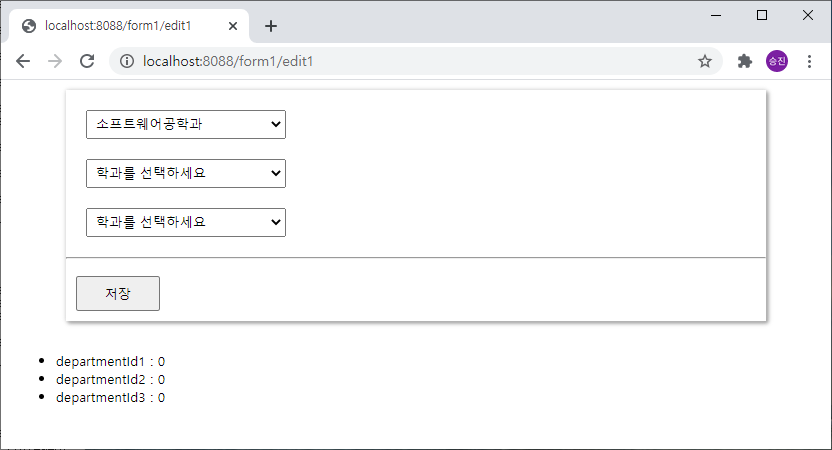
</select>

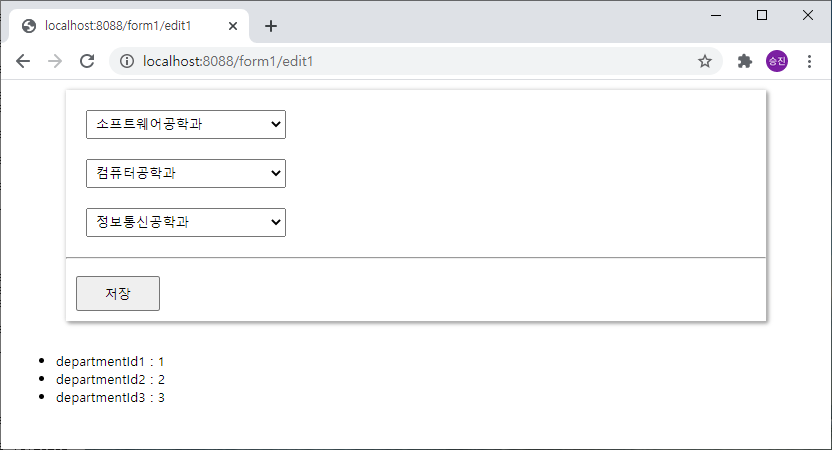
줄 26~29와

줄 32~38의 출력 결과는 같다

만약 학과를 선택하세요 라는 출력문이 필요 없으면 줄 23이 가장 간단하다.

(줄 23은 학과를 선택하세요 라는 옵션이 없음)





# radio 버튼 구현

## Form1Controller.java

### src/main/java/net/skhu/model/Form1.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.controller;  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.ModelAttribute;  import org.springframework.web.bind.annotation.PostMapping;  import net.skhu.mapper.DepartmentMapper;  import net.skhu.model.Form1;  @Controller  public class Form1Controller {  @Autowired  DepartmentMapper departmentMapper;  @ModelAttribute  public void addSomething(Model model) {  model.addAttribute("departments", departmentMapper.findAll());  }  @GetMapping("form1/edit1")  public String edit1(Model model) {  model.addAttribute("form1", new Form1());  return "form1/edit1";  }  @PostMapping("form1/edit1")  public String edit1(Model model, Form1 form1) {  return "form1/edit1";  }  @GetMapping("form1/edit2")  public String edit2(Model model) {  model.addAttribute("form1", new Form1());  return "form1/edit2";  }  @PostMapping("form1/edit2")  public String edit2(Model model, Form1 form1) {  return "form1/edit2";  }  } |

edit2 액션 메소드 구현은, edit1 액션 메소드와 거의 같다.

액션 메소드 URL과 뷰의 이름만 다르다.

## form1/edit2.jsp

### src/main/webapp/WEB-INF/views/form1/edit2.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 | <%@ 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">  <link rel="stylesheet" href="${R}common.css">  <style>  form { width: 700px; margin: auto;  box-shadow: 0 0 4px lightgray, 2px 2px 4px gray; overflow: auto; }  form div { margin: 20px; }  div.buttons label { margin-right: 20px; }  button { margin: 10px; padding: 0.5em 2em; }  ul { margin-top: 30px; }  </style>  </head>  <body>  <div class="container">  <form:form method="post" modelAttribute="form1">  <div class="buttons">  <form:radiobuttons path="departmentId1" itemValue="id"  itemLabel="shortName" items="${ departments }" />  </div>  <div class="buttons">  <form:radiobutton path="departmentId2" value="1" label="소프" />  <form:radiobutton path="departmentId2" value="2" label="컴공" />  <form:radiobutton path="departmentId2" value="3" label="정통" />  <form:radiobutton path="departmentId2" value="4" label="글티" />  </div>  <hr />  <button type="submit">저장</button>  </form:form>    <ul>  <li>departmentId1 : ${ form1.departmentId1 }</li>  <li>departmentId2 : ${ form1.departmentId2 }</li>  </ul>  </div>  </body>  </html> |

<form:radiobuttons path="departmentId1" itemValue="id" itemLabel="shortName" items="${ departments }" />

만약 form1 모델 객체의 departmentId1 속성 값이 4 이라면,

위 확장 태그의 실행 결과 출력은 다음과 같다.

<span>

<input id="departmentId11" name="departmentId1" type="radio" value="1"/>

<label for="departmentId11">소프</label>

</span>

<span>

<input id="departmentId12" name="departmentId1" type="radio" value="2"/>

<label for="departmentId12">컴공</label>

</span>

<span>

<input id="departmentId13" name="departmentId1" type="radio" value="3"/>

<label for="departmentId13">정통</label>

</span>

<span>

<input id="departmentId14" name="departmentId1" type="radio" value="4" checked />

<label for="departmentId14">글티</label>

</span>

<label for="departmentId11">소프</label>

label 태그의 for 애트리뷰트 값은 어떤 input 태그의 id 이다.

이 label 태그를 클릭한 것은 그 input 태그를 클릭한 것과 같게 된다.

그래서 정확히 radio 박스를 클릭하지 않아도, label을 클릭해도 된다.

name 값이 같은 radio 들은 하나의 그룹이된다.

그 그룹에서 오직 한 개만 선택될 수 있다.

선택된 radio의 value 값이 request parameter로 전달된다.

<form:radiobutton path="departmentId2" value="1" label="소프" />

<form:radiobutton path="departmentId2" value="2" label="컴공" />

<form:radiobutton path="departmentId2" value="3" label="정통" />

<form:radiobutton path="departmentId2" value="4" label="글티" />

만약 form1 모델 객체의 departmentId2 속성 값이 1 이라면,

위 확장 태그의 실행 결과 출력은 다음과 같다.

<input id="departmentId21" name="departmentId2" type="radio" value="1" checked />

<label for="departmentId21">소프</label>

<input id="departmentId22" name="departmentId2" type="radio" value="2"/>

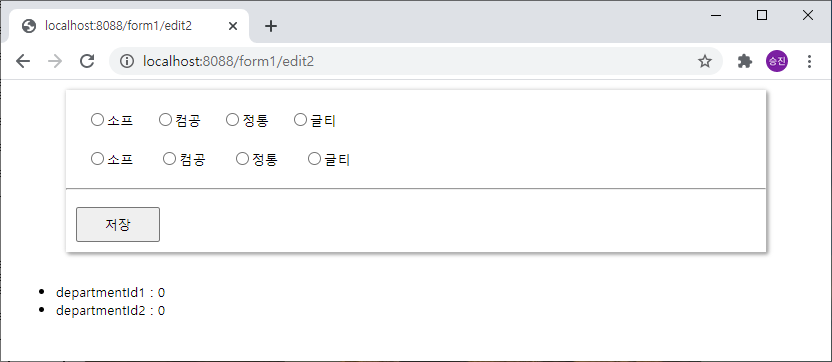
<label for="departmentId22">컴공</label>

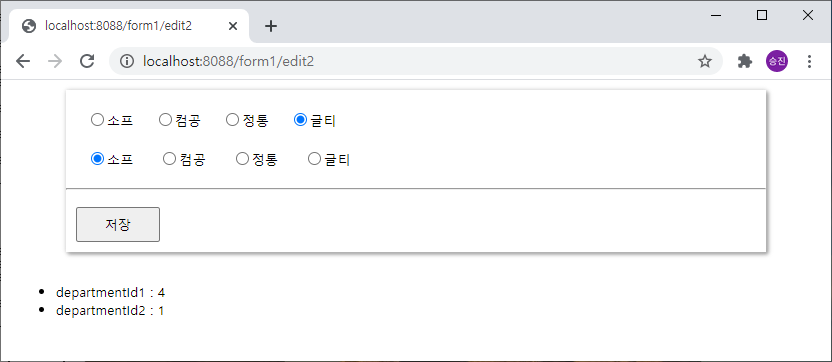
<input id="departmentId23" name="departmentId2" type="radio" value="3"/>

<label for="departmentId23">정통</label>

<input id="departmentId24" name="departmentId2" type="radio" value="4"/>

<label for="departmentId24">글티</label>





# checkbox 구현

## Form1Controller.java

### src/main/java/net/skhu/controller/Form1Controller.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 | 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.GetMapping;  import org.springframework.web.bind.annotation.ModelAttribute;  import org.springframework.web.bind.annotation.PostMapping;  import net.skhu.mapper.DepartmentMapper;  import net.skhu.model.Form1;  @Controller  public class Form1Controller {  @Autowired  DepartmentMapper departmentMapper;  @ModelAttribute  public void addSomething(Model model) {  model.addAttribute("departments", departmentMapper.findAll());  }  @GetMapping("form1/edit1")  public String edit1(Model model) {  model.addAttribute("form1", new Form1());  return "form1/edit1";  }  @PostMapping("form1/edit1")  public String edit1(Model model, Form1 form1) {  return "form1/edit1";  }  @GetMapping("form1/edit2")  public String edit2(Model model) {  model.addAttribute("form1", new Form1());  return "form1/edit2";  }  @PostMapping("form1/edit2")  public String edit2(Model model, Form1 form1) {  return "form1/edit2";  }  @GetMapping("form1/edit3")  public String edit3(Model model) {  model.addAttribute("form1", new Form1());  return "form1/edit3";  }  @PostMapping("form1/edit3")  public String edit3(Model model, Form1 form1) {  return "form1/edit3";  }  } |

edit3 액션 메소드 구현은, edit1, edit2 액션 메소드와 거의 같다.

액션 메소드 URL과 뷰의 이름만 다르다.

## form1/edit3.jsp

### src/main/webapp/WEB-INF/views/form1/edit3.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 | <%@ 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">  <link rel="stylesheet" href="${R}common.css">  <style>  form { width: 700px; margin: auto;  box-shadow: 0 0 4px lightgray, 2px 2px 4px gray; overflow: auto; }  form div { margin: 20px; }  div.buttons label { margin-right: 20px; }  button { margin: 10px; padding: 0.5em 2em; }  ul { margin-top: 30px; }  </style>  </head>  <body>  <div class="container">  <form:form method="post" modelAttribute="form1">  <div class="buttons">  <form:checkboxes path="departmentId4" itemValue="id"  itemLabel="shortName" items="${ departments }" />  </div>  <div class="buttons">  <form:checkbox path="departmentId5" value="1" label="소프" />  <form:checkbox path="departmentId5" value="2" label="컴공" />  <form:checkbox path="departmentId5" value="3" label="정통" />  <form:checkbox path="departmentId5" value="4" label="글티" />  </div>  <div>  <form:checkbox path="enabled" value="true" label="활성화" />  </div>  <hr />  <button type="submit">저장</button>  </form:form>    <ul>  <li>departmentId4 :  <c:forEach var="i" items="${form1.departmentId4}" >  ${i}  </c:forEach>  </li>  <li>departmentId5 :  <c:forEach var="i" items="${form1.departmentId5}" >  ${i}  </c:forEach>  </li>  <li>enabled: ${ form1.enabled }</li>  </ul>  </div>  </body>  </html> |

<form:checkboxes path="departmentId4" itemValue="id" itemLabel="shortName" items="${ departments }" />

만약 form1 모델 객체의 departmentId4 속성 값이 { 4 } 이면,

위 확장 태그의 실행 결과 출력은 다음과 같다.

<span>

<input id="departmentId41" name="departmentId4" type="checkbox" value="1"/>

<label for="departmentId41">소프</label>

</span>

<span>

<input id="departmentId42" name="departmentId4" type="checkbox" value="2"/>

<label for="departmentId42">컴공</label>

</span>

<span>

<input id="departmentId43" name="departmentId4" type="checkbox" value="3"/>

<label for="departmentId43">정통</label>

</span>

<span>

<input id="departmentId44" name="departmentId4" type="checkbox" value="4" checked />

<label for="departmentId44">글티</label>

</span>

name이 같은 radio들은 그 중 하나만 선택할 수 있지만,

name이 같은 checkbox들은 여러 개 체크할 수 있다.

체크된 checkbox들의 값이 request parameter가 되어 전달된다.

따라서 같은 이름으로 여러 개의 값이 request parameter로 전달된다.

이 값을 전달 받는 변수는 배열이어야 한다.

@Data

public class Form1 {

int departmentId1;

int departmentId2;

int departmentId3;

int[] departmentId4;

int[] departmentId5;

boolean enabled;

}

<form:checkbox path="departmentId5" value="1" label="소프" />

<form:checkbox path="departmentId5" value="2" label="컴공" />

<form:checkbox path="departmentId5" value="3" label="정통" />

<form:checkbox path="departmentId5" value="4" label="글티" />

만약 form1 모델 객체의 departmentId5 속성 값이 { 1, 3 } 배열이면,

위 확장 태그의 실행 결과 출력은 다음과 같다.

<input id="departmentId51" name="departmentId5" type="checkbox" value="1" checked />

<label for="departmentId51">소프</label>

<input id="departmentId52" name="departmentId5" type="checkbox" value="2"/>

<label for="departmentId52">컴공</label>

<input id="departmentId53" name="departmentId5" type="checkbox" value="3" checked />

<label for="departmentId53">정통</label>

<input id="departmentId54" name="departmentId5" type="checkbox" value="4"/>

<label for="departmentId54">글티</label>

<form:checkbox path="enabled" value="true" label="활성화" />

이 체크 박스가 체크되면, 이름=enable, 값="true" request parameter가 서버에 전달된다.

이 값은 boolean 타입의 변수에 대입된다.

@Data

public class Form1 {

int departmentId1;

int departmentId2;

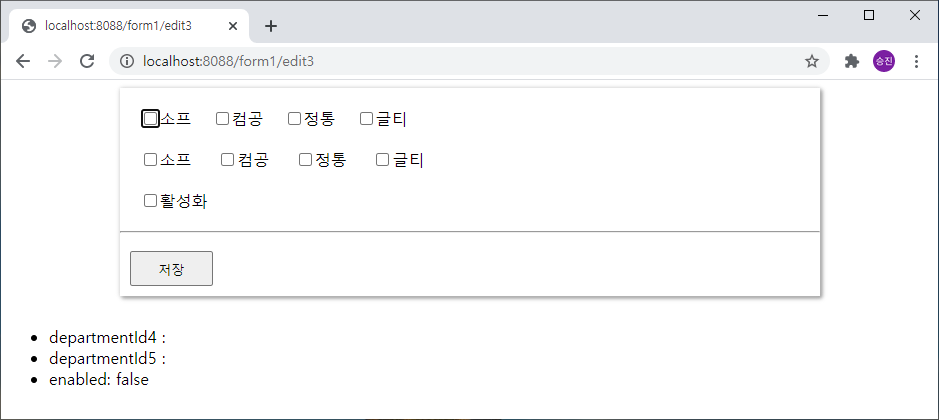
int departmentId3;

int[] departmentId4;

int[] departmentId5;

boolean enabled;

}



Graphical user interface, text, application, email

Description automatically generated