입력 폼 #1

2021-10-11

이승진

**학습목표**

입력 폼을 구현한다

텍스트 입력

목록 선택

체크 박스

라디오 버튼

**목차**

[1. HTTP Request, HTTP Response 2](#_Toc68295996)

[1) HTTP Request란? 2](#_Toc68295997)

[2) HTTP Request 구조 2](#_Toc68295998)

[2. request parameter 전달 3](#_Toc68295999)

[1) request parameter란? 3](#_Toc68296000)

[2) request parameter 종류 3](#_Toc68296001)

[3. request method 4](#_Toc68296002)

[1) request method란? 4](#_Toc68296003)

[2) request method 종류 4](#_Toc68296004)

[3) request method 지정 방법 5](#_Toc68296005)

[4) request method와 request parameter 5](#_Toc68296006)

[5) 액션 메소드의 어노테이션 5](#_Toc68296007)

[4. request1 프로젝트 6](#_Toc68296008)

[1) 프로젝트 생성 6](#_Toc68296009)

[2) pom.xml 수정 9](#_Toc68296010)

[3) application.properties 수정 10](#_Toc68296011)

[5. form/text1 페이지 구현 11](#_Toc68296012)

[1) 개요 11](#_Toc68296013)

[2) Form1Controller.java 생성 11](#_Toc68296014)

[3) form1/text1.jsp 생성 14](#_Toc68296015)

[4) 실행 19](#_Toc68296016)

[6. form1/text2 페이지 구현 22](#_Toc68296017)

[1) Form1Controller.java 수정 22](#_Toc68296018)

[2) form1/text2.jsp 생성 23](#_Toc68296019)

[3) 실행 24](#_Toc68296020)

[7. form2/select1 페이지 구현 29](#_Toc68296021)

[1) Form2Controller.java 생성 29](#_Toc68296022)

[2) form2/select1.jsp 생성 30](#_Toc68296023)

[3) 실행 31](#_Toc68296024)

[8. form2/select2 페이지 구현 32](#_Toc68296025)

[1) Form2Controller.java 수정 32](#_Toc68296026)

[2) form2/select2.jsp 생성 33](#_Toc68296027)

[9. form3/check1 페이지 구현 35](#_Toc68296028)

[1) Form3Controller.java 생성 35](#_Toc68296029)

[2) form3/check1.jsp 생성 36](#_Toc68296030)

[3) 실행 37](#_Toc68296031)

[10. 연습문제 38](#_Toc68296032)

# HTTP Request, HTTP Response

## HTTP Request란?

웹브라우저와 웹서버는 서로 데이터를 주고 받으며 통신한다.

웹브라우저로부터 웹서버로 전달되는 데이터를 HTTP Request 이라고 한다.

웹브라우저가 웹서버에게 무엇인가를 요구(요청)하는 것이 HTTP Request 이다.

웹서버로부터 웹브라우저에게 전달되는 데이터를 HTTP Response 이라고 한다.

웹브라우저로부터 요구(요청) 받은 것을 처리한 결과 데이터가 HTTP Response 이다.

URI=Unified Resourse id

URL의 상위개념

## HTTP Request 구조

### URL(Unified Resourse Locator)

요청하는 대상에 대한 주소이다.

예: https://www.google.com/search?q=student

### HTTP Request Method

이 요청이 어떤 종류의 작업을 서버에 요청하는 것인지 구분하기 위한 속성이다.

예: GET, POST

### HTTP Request Header

요청에 대한 부가 설명이 들어있다.

예: 웹브라우저 버전, 웹브라우저 언어 설정, 문자 인코딩

### HTTP Request Body

요청에 포함된 데이터가 들어있다.

예: 파일 업로드의 경우 파일 내용이 들어있다.

게시글 저장 요청의 경우, 게시글 제목과 내용이 들어있다.

# request parameter 전달

## request parameter란?

웹브라우저가 웹서버의 URL을 요청하는 것을 HTTP request이라고 한다.

서버가 Spring Web MVC 앱인 경우, 이 요청에 의해서 컨트롤러의 액션 메소드가 자동으로 호출된다.

이 요청(HTTP Request)에 같이 담겨서 서버에 전달되는 파라미터를, request parameter이라고 한다.

request parameter는 액션 메소드의 파라미터 변수에 전달된다.

## request parameter 종류

다음과 같은 것들이 request parameter가 되어 서버에 전달된다.

### query string

외울 것(Request parameter)

URL의 query string은 request parameter가 되어 서버에 전달된다.

예:

|  |
| --- |
| http://localhost:8088/first?**id=3&name=tom** |

?뒤의 id name값

이 URL에 의해서 다음과 같은 request parameter가 서버에 전달된다.

|  |  |
| --- | --- |
| request parameter name | request parameter value |
| id | 3 |
| name | tom |

### 입력 폼에 입력된 값

|  |
| --- |
| <form method="post">  <input type="text" name="**title**" />  <input type="number" name="**price**"/>  <button type="submit">저장</button>  </form> |

위 form 태그에 의해, 웹브라우저창에 입력폼이 그려진다.

이 입력폼의 두 칸에 '우유', '1000'을 각각 입력하고 submit 버튼을 클릭하면,

다음과 같은 request parameter가 서버에 전달된다.

input 태그의 name 속성값이 request parameter name이 된다.

|  |  |
| --- | --- |
| request parameter name | request parameter value |
| title | 우유 |
| price | 1000 |

# request method

## request method란?

웹브라우저가 웹서버의 URL을 요청하는 것을 HTTP request이라고 한다.

HTTP request에는 method 속성이 있다.

여기서 method는 객체지향 언어의 method 와는 다른 것이다.

HTTP request의 method는, 이 요청이 어떤 종류의 작업을 서버에 요청하는 것인지 구분하기 위한 속성이다.

HTTP request의 method를, **요청 방식** 정도로 해석하자.

## request method 종류

HTTP request method 속성의 값의 종류는 다음과 같다.

|  |  |
| --- | --- |
| 값 | 설명 |
| GET | (조회 요청) 서버의 어떤 자료를 조회하기 위한 요청 |
| POST | (등록/수정/삭제 요청) 자료를 등록/수정/삭제하기 위한 요청 |

등록/수정/삭제 요청 마다 request method를 구분하여 사용하는 것이 바람직하지만,

초급 웹프로그래밍에서는 GET 과 POST 만 사용하는 것이 일반적이다.

### 참고

|  |  |
| --- | --- |
| 값 | 설명 |
| GET | (조회 요청) 서버의 어떤 자료를 조회하기 위한 요청 |
| POST | (등록 요청) 새 자료를 등록 위한 요청 |
| PUT | (수정 요청) 기존 자료를 수정하기 위한 요청 |
| DELETE | (삭제 요청) 기존 자료를 삭제하기 위한 요청 |

고급 웹프로그래밍에서 위와 같이 구분하여 사용한다.

이 수업에서는 GET / POST 만 사용한다.

## request method 지정 방법

### form 태그

예:

|  |
| --- |
| <form method="POST"> |

위의 예와 같이 form 태그에 request method를 지정할 수 있다.

위 form 태그 내부의 submit 버튼이 클릭되면, POST 방식의 HTTP request가 서버에 전달된다.

form 태그에 request method를 지정하지 않으면, 디폴트 값은 GET 이다.

POST와 GET의 구별은 URL을 보면 알 수 있다.

(GET방식의 경우 URL에 입력 값이 보인다)

POST방식은 body에도 들어가 있고 queryString에도 들어가 있다

GET은 body가 존재하지 않고 queryString에만 붙어있다.

### 디폴트 request method는 GET 이다.

request method를 따로 지정하지 않으면, 디폴트 값은 GET 이다.

- a 태그가 클릭되어 서버에 HTTP request가 전달될 때.

- 웹브라우저의 주소칸에 URL을 입력하고 엔터키를 눌렀을 때.

- 즐겨찾기 항목을 눌렀을 때.

## request method와 request parameter

### GET

GET 요청에는 HTTP request body 부분이 없다.

따라서 GET 방식 요청일 때, request parameter는 URL의 query string 뿐이다.

GET 방식의 입력 폼에 입력된 값들은,

그 폼의 submit 버튼이 눌려져 서버에 HTTP request가 전달될 때,

URL의 query string에 자동으로 붙어서 서버에 전달된다.

### POST

POST 요청일 경우에, request parameter 다음과 같다.

- URL의 query string

- HTTP request body 에 들어있는 데이터

POST 방식 입력 폼에 입력된 값들은,

그 폼의 submit 버튼이 눌려져, HTTP request가 서버에 전달될 때,

HTTP request body에 담겨서 서버에 전달된다.

## 액션 메소드의 어노테이션

액션 메소드에는 다음의 어노테이션 중 하나가 붙어있어야 한다.

@RequestMapping("*URL*") - *URL*이 요청된 경우에 이 액션 메소드가 실행된다. (GET, POST 무관)

@GetMapping("*URL*") - *URL*이 GET 방식으로 요청된 경우에만 이 액션 메소드가 실행된다

@PostMapping("*URL*") - *URL*이 POST 방식으로 요청된 경우에만 이 액션 메소드가 실행된다

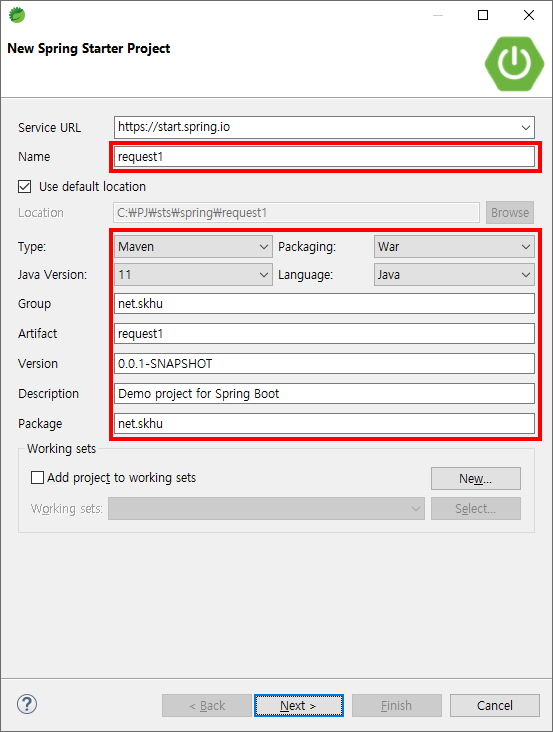
액션 메소드에 위 어노테이션 중 하나가 붙어있지 않거나

*URL*에 오타가 있으면, *URL*을 요청했을 때 404 Not Found 에러가 발생한다.

# request1 프로젝트

## 프로젝트 생성

메뉴: File - New - Spring Starter Project



|  |  |
| --- | --- |
| Name | request1 |
| Group | net.skhu |
| Artifact | request1 |
| Package | net.skhu |

Graphical user interface, text, application, email

Description automatically generated

Developer Tools 그룹 아래에서

Spring Boot DevTools 항목 체크

Graphical user interface, text, application, email

Description automatically generated

Web 그룹 아래에서

Spring Web 항목 체크

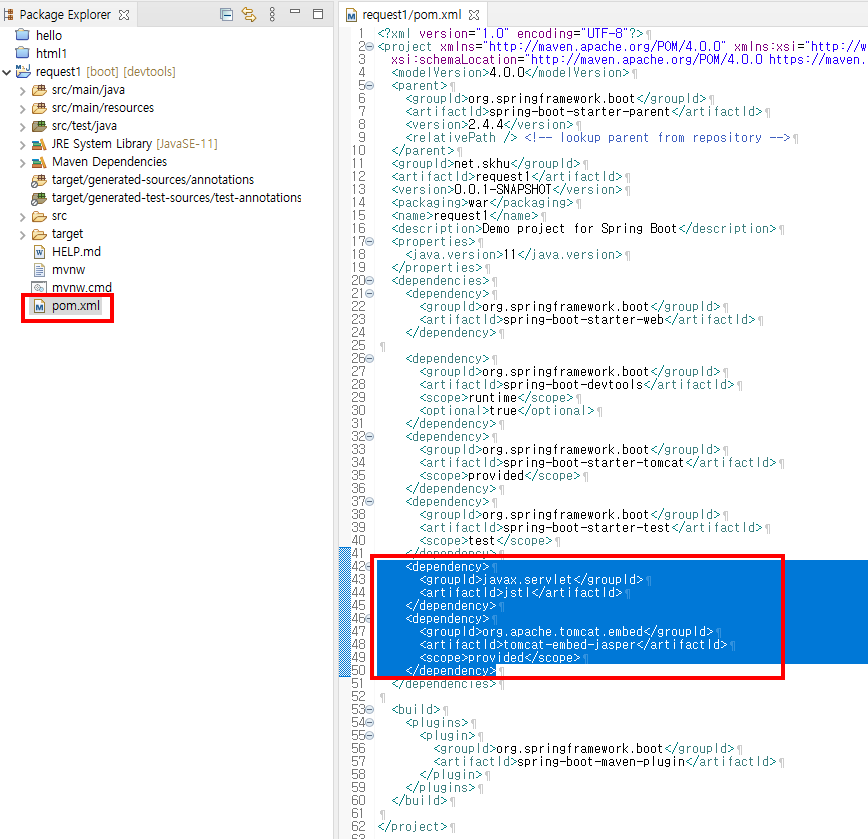
위와 같이 선택하고

Finish 버튼 클릭

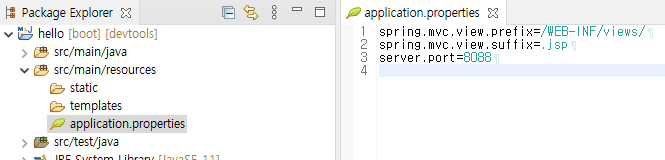
## pom.xml 수정

|  |
| --- |
| <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  </dependency>  <dependency>  <groupId>org.apache.tomcat.embed</groupId>  <artifactId>tomcat-embed-jasper</artifactId>  <scope>provided</scope>  </dependency> |

위 내용을 <dependencies> 태그 내부에 추가한다.



## application.properties 수정



### src/main/resources/application.properties

|  |
| --- |
| spring.mvc.view.prefix=/WEB-INF/views/  spring.mvc.view.suffix=.jsp  server.port=8088 |

spring.mvc.view.prefix=/WEB-INF/views/

spring.mvc.view.suffix=.jsp

뷰(view) 파일의 위치와 확장자를 지정한다.

/WEB-INF/views/뷰이름.jsp

서버가 사용할 포트 번호를 8088로 설정한다.

위 설정을 하지 않으면, 디폴트 포트 번호는 8080 이다.

8080 포트를 사용하는 앱들이 많아서 포트 충돌 에러가 발생할 확률이 높으니,

8088로 수정하는 것이 바람직하다.

Views/,jsp 뒤에 공백이 들어가면 안 된다 공백이 들어가면 404 Not Found 오류가 발생할 수 있다.

# form/text1 페이지 구현

## 개요

웹브라우저가 http://localhost:8088/form/text1 URL을 서버에 요청했을 때,

서버에서 실행될 웹 페이지를 구현하자.

### controller 클래스 구현

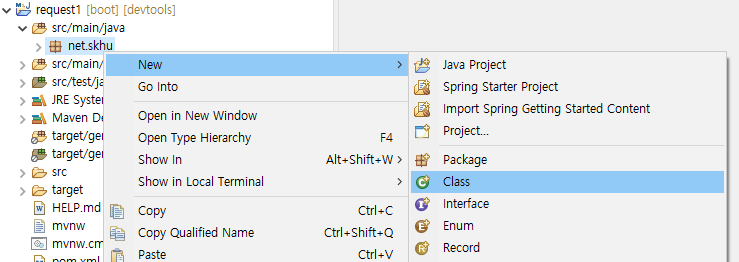
URL 요청에 의해서 서버에서 실행될 controller 클래스와 action method를 구현해야 한다.

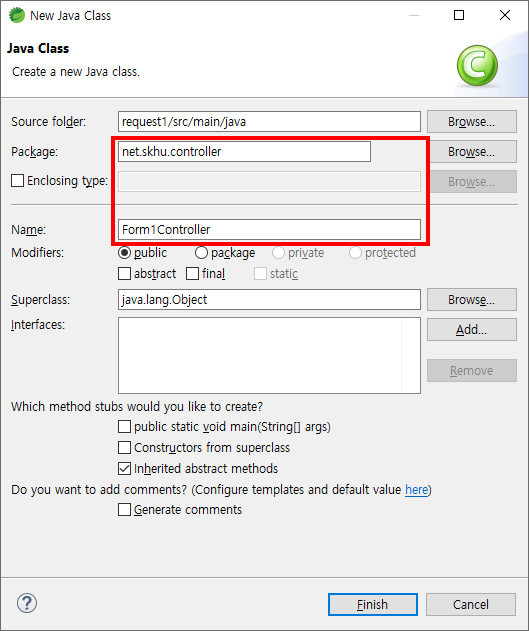
### view 파일 구현

home/index 페이지의 내용을 출력하기 뷰 파일을 구현해야 한다.

텍스트를 입력하는 입력폼을 구현해 보자.

## Form1Controller.java 생성





|  |  |
| --- | --- |
| Package: | net.skhu.controller |
| Name: | Form1Controller |

### 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 | package net.skhu.controller;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PostMapping;  @Controller  public class Form1Controller {  @GetMapping("form1/text1")  public String text1(Model model) {  return "form1/text1";  }  @PostMapping("form1/text1")  public String text1(Model model, String param1, String param2) {  System.out.println(param1);  System.out.println(param2);  return "form1/text1";  }  } |

@Controller

컨트롤러 클래스에는 이 어노테이션이 붙어있어야 한다.

그렇지 않으면 404 Not Found 에러가 발생한다.

@GetMapping("form1/text1")

public String text1(Model model) {

"http://서버주소:포트번호/form1/text1" URL이 GET 방식으로 서버에 요청되면,

이 액션 메소드가 자동으로 호출되어 실행된다.

@PostMapping("form1/text1")

public String text1(Model model, String param1, String param2) {

"http://서버주소:포트번호/form1/text1" URL이 POST 방식으로 서버에 요청되면,

이 액션 메소드가 자동으로 호출되어 실행된다.

### 실행순서

(1) 웹브라우저 주소칸에 "http://서버주소:포트번호/form1/text1" URL을 입력하고 엔터키를 누르면,

이 URL이 서버에 GET 방식으로 요청된다.

(2) @GetMapping(...) 어노테이션이 붙은 text1 액션 메소드가 자동으로 호출된다.

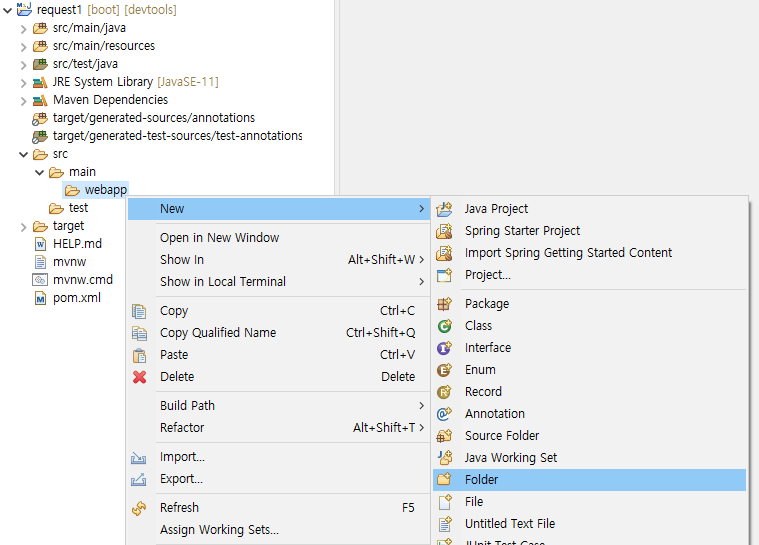
(3) 이 액션 메소드는 "form1/text1" 뷰 이름을 리턴한다.

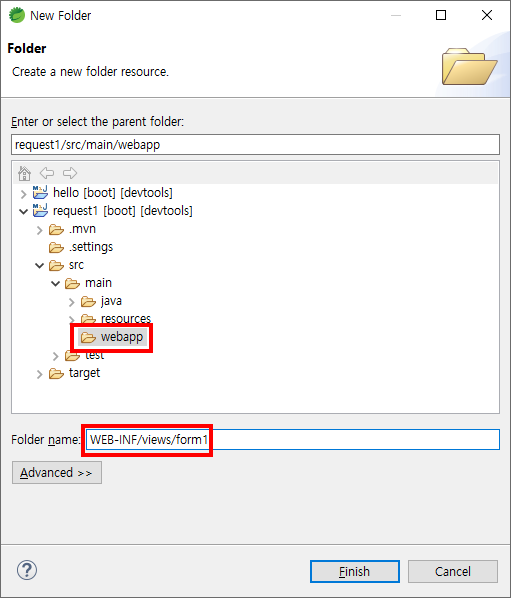
(4) /WEB-INF/views/form1/test1.jsp 뷰 파일이 실행된다.

이 뷰파일의 실행 결과 출력이 웹브라우저로 전송된다.

## form1/text1.jsp 생성

### WEB-INF/views/form1 폴더 생성





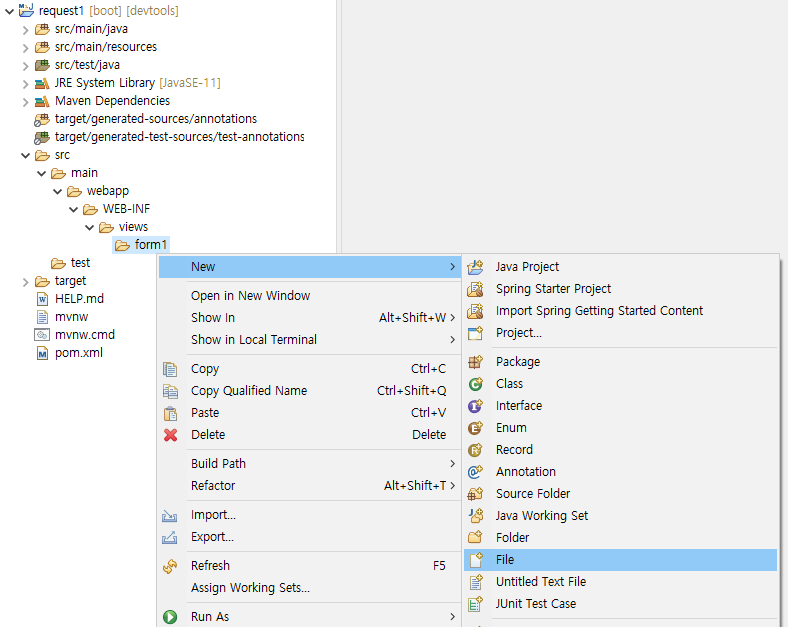
|  |  |
| --- | --- |
| Folder name: | WEB-INF/views/form1 |

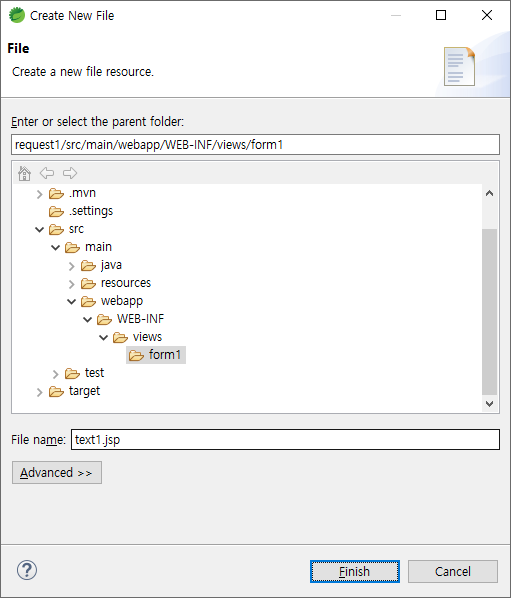
### 주의

위 폴더 경로명에서 **WEB-INF/views/** 부분은

application.properties 파일의 spring.mvc.view.prefix 항목과 일치해야 한다.

일치하지 않을 경우에 404 Not Found 에러가 발생한다.





|  |  |
| --- | --- |
| File name: | text1.jsp |

### src/main/webapp/WEB-INF/views/form1/text1.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 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <meta name="viewport" content="width=device-width, initial-scale=1">  <style>  div.container { width: 600px; margin: 20px auto; }  form div { margin-bottom: 10px; }  input { width: 200px; padding: 4px; }  button { padding: 0.4em 3em; margin-top: 10px; }  </style>  </head>  <body>  <div class="container">  <form method="post">  <h1>텍스트 입력폼 01</h1>  <div>  <label>text1:</label>  <input type="text" name="param1" value="아무 것이나 입력하시오1" />  </div>  <div>  <label>text2:</label>  <input type="text" name="param2" value="아무 것이나 입력하시오2" />  </div>    <button type="submit">확인</button>  <button type="reset">취소</button>  </form>  </div>  </body>  </html> |

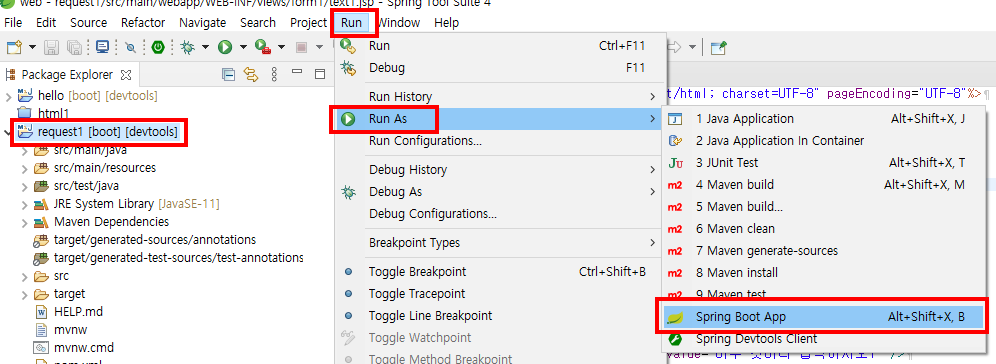
<input type="text" name="param1" value="아무 것이나 입력하시오1" />

input 태그가 웹브라우저창에 그려질 때, value 속성값이 채워진 상태로 그려진다.

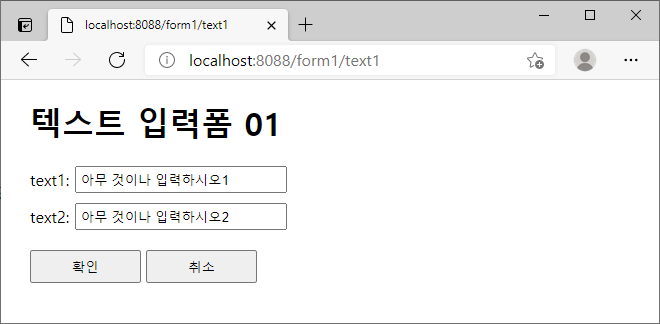
value 속성값이 "아무 것이나 입력하시오1" 상수이므로,

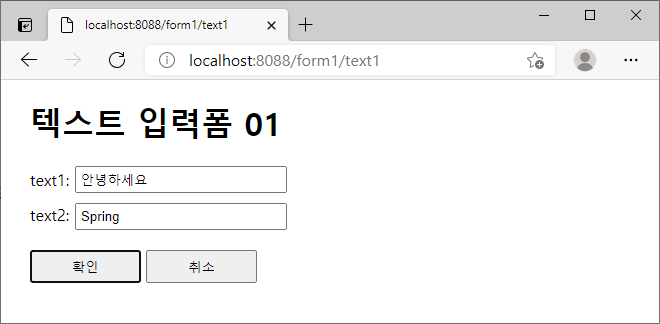
이 input 태그는 언제나 이 값이 채워진 상태로 그려질 것이다.

## 실행



http://localhost:8088/form1/text1





위 화면처럼 입력하고 확인 버튼을 클릭하자.

<form method="post">

위와 같이 정의된 form 태그 내부의 type="submit" 버튼이 클릭되었으므로,

현재 웹브라우저 주소칸의 URL이 서버에 다시 요청된다.

http://localhost:8088/form1/text1

POST 방식으로 요청된다.

이때 입력 폼의 input 태그에 입력된 값도, 요청에 담겨서 서버에 전달된다.

이렇게 서버에 전달되는 값을 request parameter 이라고 한다.

|  |  |
| --- | --- |
| request parameter 이름 | request paramter 값 |
| param1 | 안녕하세요 |
| param2 | Spring |

|  |
| --- |
| @PostMapping("form1/text1")  public String text1(Model model, String param1, String param2) {  System.out.println(param1);  System.out.println(param2);  return "form1/text1";  } |

http://localhost:8088/form1/text1 URL이 POST 방식으로 요청되었으므로

이 메소드가 자동으로 호출된다.

위 액션 메소드의 파라미터 변수에 request paramter 값이 채워져서 전달된다.

param1 이름의 request paramter 값이 String param1 파라미터 변수에 채워져서 전달되고

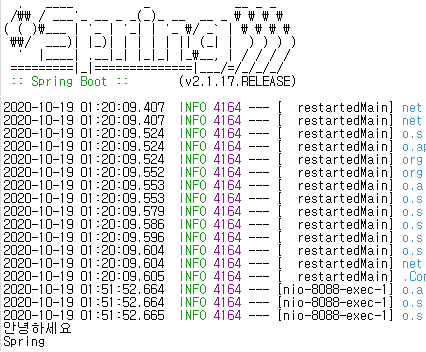
param2 이름의 request paramter 값이 String param2 파라미터 변수에 채워져서 전달된다.

즉 request parameter 이름과 같은 이름의 액션 메소드 파라미터 변수에 값이 채워진다.

System.out.println(param1);

System.out.println(param2);

그렇게 전달된 값은 eclipse 콘솔에 출력된다.



이 액션 메소드는 "form1/text1" 뷰 이름을 리턴한다.

WEB-INF/views/form1/text1.jsp 뷰 파일이 다시 실행되고

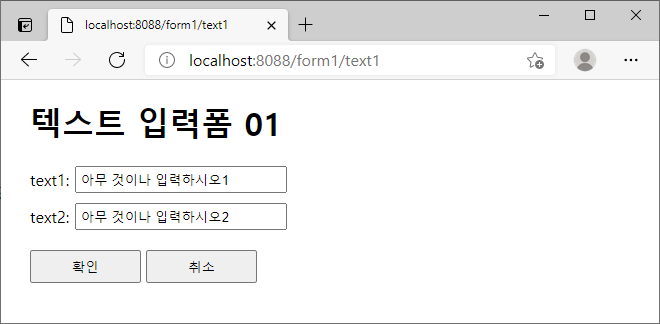
이 뷰 파일의 실행 결과 출력이 웹브라우저로 출력된다.

<input type="text" name="param1" value="아무 것이나 입력하시오1" />

input 태그가 웹브라우저창에 그려질 때, value 속성값이 채워진 상태로 그려진다.

value 속성값이 "아무 것이나 입력하시오1" 상수이므로,

이 input 태그는 언제나 이 값이 채워진 상태로 그려질 것이다.



서버에서 웹브라우저로 출력된 내용은 다음과 같다.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <meta name="viewport" content="width=device-width, initial-scale=1">  <style>  div.container { width: 600px; margin: 20px auto; }  form div { margin-bottom: 10px; }  input { width: 200px; padding: 4px; }  button { padding: 0.4em 3em; margin-top: 10px; }  </style>  </head>  <body>  <div class="container">  <form method="post">  <h1>텍스트 입력폼 01</h1>  <div>  <label>text1:</label>  <input type="text" name="param1" value="아무 것이나 입력하시오1" />  </div>  <div>  <label>text2:</label>  <input type="text" name="param2" value="아무 것이나 입력하시오2" />  </div>    <button type="submit">확인</button>  <button type="reset">취소</button>  </form>  </div>  </body>  </html> |

# form1/text2 페이지 구현

## Form1Controller.java 수정

### 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 | package net.skhu.controller;  import java.util.Date;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PostMapping;  @Controller  public class Form1Controller {  @GetMapping("form1/text1")  public String text1(Model model) {  return "form1/text1";  }  @PostMapping("form1/text1")  public String text1(Model model, String param1, String param2) {  System.out.println(param1);  System.out.println(param2);  return "form1/text1";  }  @GetMapping("form1/text2")  public String text2(Model model) {  model.addAttribute("param1", "hello");  model.addAttribute("param2", "world");  model.addAttribute("now", new Date());  return "form1/text2";  }  @PostMapping("form1/text2")  public String text2(Model model, String param1, String param2) {  System.out.println(param1);  System.out.println(param2);  System.out.println(new Date());  model.addAttribute("param1", param1);  model.addAttribute("param2", param2);  model.addAttribute("now", new Date());  return "form1/text2";  }  } |

## form1/text2.jsp 생성

### src/main/webapp/WEB-INF/views/form1/text2.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 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ taglib prefix = "fmt" uri = "http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <meta name="viewport" content="width=device-width, initial-scale=1">  <style>  div.container { width: 600px; margin: 20px auto; }  form div { margin-bottom: 10px; }  input { width: 200px; padding: 4px; }  button { padding: 0.4em 3em; margin-top: 10px; }  </style>  </head>  <body>  <div class="container">  <form method="post">  <h1>텍스트 입력폼 02</h1>  <div>  <label>text1:</label>  <input type="text" name="param1" value="${ param1 }" />  </div>  <div>  <label>text2:</label>  <input type="text" name="param2" value="${ param2 }" />  </div>    <button type="submit">확인</button>  <button type="reset">취소</button>  </form>  <p> <fmt:formatDate pattern = "yyyy-MM-dd HH:mm:ss" value = "${ now }" /> </p>  </div>  </body>  </html> |

## 실행

(1) 웹브라우저에서 http://localhost:8088/form1/text2 URL을 실행하자.

(2) 웹브라우저는 위 URL을 GET 방식으로 서버에 요청한다.

(3) 아래 메소드가 서버에서 실행된다.

|  |
| --- |
| @GetMapping("form1/text2")  public String text2(Model model) {  model.addAttribute("param1", "hello");  model.addAttribute("param2", "world");  model.addAttribute("now", new Date());  return "form1/text2";  } |

모델(model)에 다음과 같은 데이터가 채워진다.

|  |  |
| --- | --- |
| model attribute 이름 | model attribute 값 |
| param1 | hello |
| param2 | world |
| now | *현재시각* |

(4) /WEB-INF/views/form1/text2.jsp 뷰 파일이 실행된다.

<input type="text" name="param1" value="${ param1 }" />

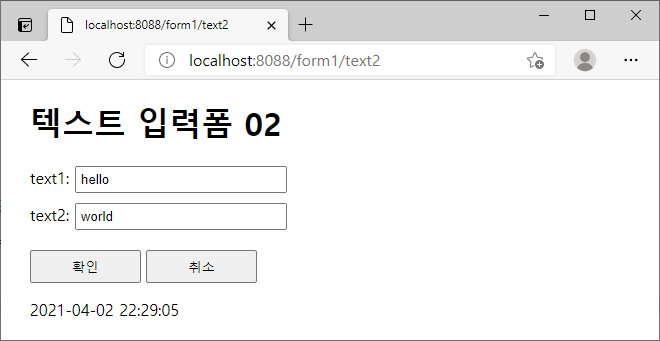
이 input 태그의 value="..." 부분에 출력되는 값은 hello 문자열이다.

<input type="text" name="param2" value="${ param2 }" />

이 input 태그의 value="..." 부분에 출력되는 값은 world 문자열이다.

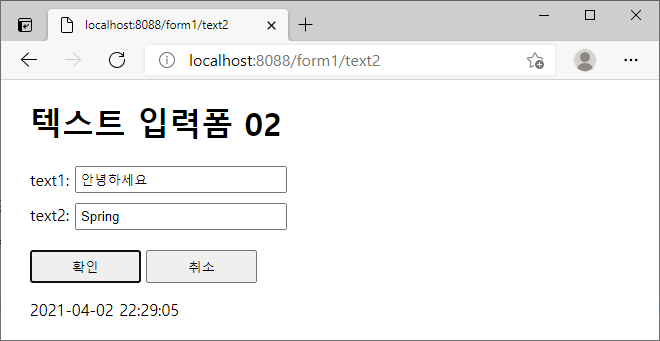
<fmt:formatDate pattern = "yyyy-MM-dd HH:mm:ss" value = "${ now }" />

서버에서 실행되는 순간의 시각이 출력된다.



서버에서 웹브라우저로 출력된 내용은 다음과 같다.

|  |
| --- |
| <%@ taglib prefix = "fmt" uri = "http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <meta name="viewport" content="width=device-width, initial-scale=1">  <style>  div.container { width: 600px; margin: 20px auto; }  form div { margin-bottom: 10px; }  input { width: 200px; padding: 4px; }  button { padding: 0.4em 3em; margin-top: 10px; }  </style>  </head>  <body>  <div class="container">  <form method="post">  <h1>텍스트 입력폼 02</h1>  <div>  <label>text1:</label>  <input type="text" name="param1" value="hello" />  </div>  <div>  <label>text2:</label>  <input type="text" name="param2" value="world" />  </div>    <button type="submit">확인</button>  <button type="reset">취소</button>  </form>  <p> 2021-04-02 22:29:05 </p>  </div>  </body>  </html> |



위와 같이 입력하고 확인 버튼을 클릭하자.

<form method="post">

위와 같이 정의된 form 태그 내부의 type="submit" 버튼이 클릭되었으므로,

현재 웹브라우저 주소칸의 URL이 서버에 다시 요청된다.

http://localhost:8088/form1/text2

POST 방식으로 요청된다.

이때 입력 폼의 input 태그에 입력된 값도, 요청에 담겨서 서버에 전달된다.

이렇게 서버에 전달되는 값을 request parameter 이라고 한다.

|  |  |
| --- | --- |
| request parameter 이름 | request paramter 값 |
| param1 | 안녕하세요 |
| param2 | Spring |

|  |
| --- |
| @PostMapping("form1/text2")  public String text2(Model model, String param1, String param2) {  System.out.println(param1);  System.out.println(param2);  System.out.println(new Date());  model.addAttribute("param1", param1);  model.addAttribute("param2", param2);  model.addAttribute("now", new Date());  return "form1/text2";  } |

http://localhost:8088/form1/text2 URL이 POST 방식으로 요청되었으므로

이 메소드가 자동으로 호출된다.

위 액션 메소드의 파라미터 변수에 request paramter 값이 채워져서 전달된다.

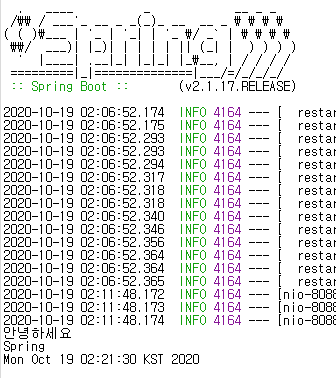
param1 이름의 request paramter 값이 String param1 파라미터 변수에 채워져서 전달되고

param2 이름의 request paramter 값이 String param2 파라미터 변수에 채워져서 전달된다.

System.out.println(param1);

System.out.println(param2);

그렇게 전달된 값은 eclipse 콘솔에 출력된다.



model.addAttribute("param1", param1);

model.addAttribute("param2", param2);

model.addAttribute("now", new Date());

위 코드에 의해서 다음과 같은 값이 모델에 채워져 뷰에 전달된다.

|  |  |
| --- | --- |
| model attribute 이름 | model attribute 값 |
| param1 | 안녕하세요 |
| param2 | Spring |
| now | *현재시각* |

now model attribute 값은 위 코드를 실행할 당시의 현재 시각이다.

이 액션 메소드는 "form1/text2" 뷰 이름을 리턴한다.

WEB-INF/views/form1/text2.jsp 뷰 파일이 다시 실행되고

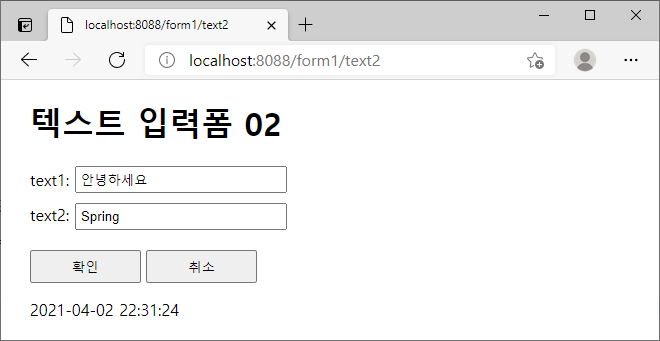
이 뷰 파일의 실행 결과 출력이 웹브라우저로 출력된다.

<input type="text" name="param1" value="${ param1 }" />

input 태그의 value="..." 부분에 출력되는 값은 안녕하세요 문자열이다.

<input type="text" name="param2" value="${ param2 }" />

input 태그의 value="..." 부분에 출력되는 값은 Spring 문자열이다.



서버에서 웹브라우저로 출력된 내용은 다음과 같다.

|  |
| --- |
| <%@ taglib prefix = "fmt" uri = "http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <meta name="viewport" content="width=device-width, initial-scale=1">  <style>  div.container { width: 600px; margin: 20px auto; }  form div { margin-bottom: 10px; }  input { width: 200px; padding: 4px; }  button { padding: 0.4em 3em; margin-top: 10px; }  </style>  </head>  <body>  <div class="container">  <form method="post">  <h1>텍스트 입력폼 02</h1>  <div>  <label>text1:</label>  <input type="text" name="param1" value="안녕하세요" />  </div>  <div>  <label>text2:</label>  <input type="text" name="param2" value="Spring" />  </div>    <button type="submit">확인</button>  <button type="reset">취소</button>  </form>  <p> 2021-04-02 22:31:24 </p>  </div>  </body>  </html> |

# form2/select1 페이지 구현

## Form2Controller.java 생성

### src/main/java/net/skhu/controller/Form2Controller.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 | package net.skhu.controller;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PostMapping;  @Controller  public class Form2Controller {  @GetMapping("form2/select1")  public String select1(Model model) {  model.addAttribute("number1", 0);  model.addAttribute("number2", 0);  model.addAttribute("cmd", "+");  model.addAttribute("result", 0);  return "form2/select1";  }  @PostMapping("form2/select1")  public String select1(Model model, Integer number1, String cmd, Integer number2) {  model.addAttribute("number1", number1);  model.addAttribute("number2", number2);  model.addAttribute("cmd", cmd);  model.addAttribute("result", calc(number1, cmd, number2));  return "form2/select1";  }  private int calc(int number1, String cmd, int number2) {  int result = 0;  switch (cmd) {  case "+": result = number1 + number2; break;  case "-": result = number1 - number2; break;  case "\*": result = number1 \* number2; break;  case "/": result = number1 / number2; break;  }  return result;  }  } |

@PostMapping("form2/select1")

public String select1(Model model, Integer number1, String cmd, Integer number2) {

이 액션 메소드의 number1, cmd, number2 파라미터 변수 각각에

number1, cmd, number2 이름의 request parameter 값이 대입되어 전달된다.

number1, number2 값은 정수이기 때문에 Integer 타입으로 선언하였다.

액션 메소드의 파라미터 변수가 기본 자료형 int 타입으로 선언될 경우에는

어떤 어노테이션을 붙여 주어야 한다. 이 어노테이션에 대해서는 나중에 설명한다.

액션 메소드의 파라미터 변수가 Integer, Double 클래스 타입인 경우에는

어떤 어노테이션을 붙여 줄 필요가 없다.

## form2/select1.jsp 생성

### src/main/webapp/WEB-INF/views/form2/select1.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 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <style>  div.container { width: 400px; margin: 20px auto; }  form div { margin-bottom: 10px; }  label { display: inline-block; width: 70px; text-align: right; }  input { width: 150px; padding: 4px; }  select { width: 50px; padding: 4px; }  button { padding: 0.4em 2em; margin-top: 10px; }  </style>  </head>  <body>  <div class="container">  <form method="post">  <h1>select 01</h1>  <div>  <label>number1:</label>  <input type="text" name="number1" value="${ number1 }" />  </div>    <div>  <label>operator:</label>  <select name="cmd">  <option value="+">+</option>  <option value="-">-</option>  <option value="\*">\*</option>  <option value="/" selected>/</option>  </select>  </div>  <div>  <label>number2:</label>  <input type="text" name="number2" value="${ number2 }" />  </div>  <div>  <button type="submit">확인</button>  </div>  </form>    <p>  결과: ${ result }  <p>  </div>  </body>  </html> |

|  |
| --- |
| <select name="cmd">  <option value="+">+</option>  <option value="-">-</option>  <option value="\*">\*</option>  <option value="/" selected>/</option>  </select> |

웹브라우저 창에 선택 목록이 그려질 때, selected 속성이 있는 option 항목이 선택된 상태로 그려진다.

따라서 이 선택 목록은 언제나 / 항목이 선택된 상태로 그려질 것이다.

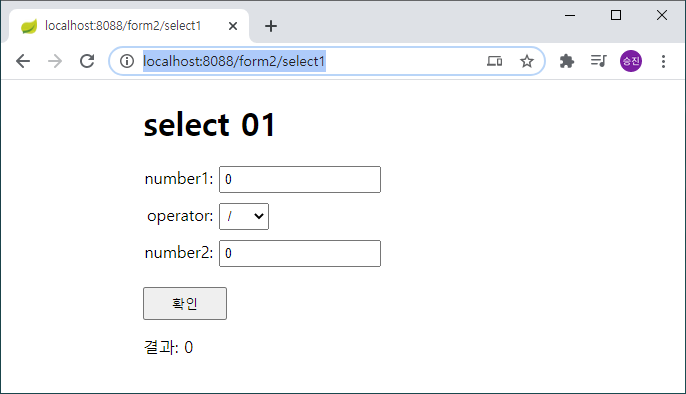
## 실행

http://localhost:8088/form2/select1

이 URL을 웹브라우저 창에서 처음 실행하면 아래 메소드가 호출된다.

|  |
| --- |
| @GetMapping("form2/select1")  public String select1(Model model) {  model.addAttribute("number1", 0);  model.addAttribute("number2", 0);  model.addAttribute("cmd", "+");  model.addAttribute("result", 0);  return "form2/select1";  } |

그 다음 WEB-INF/views/form2/select1.jsp 파일이 실행된다.



<input type="text" name="number1" value="${ number1 }" />

<input type="text" name="number2" value="${ number2 }" />

결과: ${ result }

number1, number2, result 값은 위와 같이 model attribuet 값이 출력된다.

따라서 모두 0 이 출력된다.

그런데 select 태그는 언제나 / 항목이 선택된 상태로 출력된다.

/ 항목의 option 태그에만 select 속성이 있기 때문이다.

# form2/select2 페이지 구현

## Form2Controller.java 수정

### src/main/java/net/skhu/controller/Form2Controller.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 | package net.skhu.controller;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PostMapping;  @Controller  public class Form2Controller {  @GetMapping("form2/select1")  public String select1(Model model) {  model.addAttribute("number1", 0);  model.addAttribute("number2", 0);  model.addAttribute("cmd", "+");  model.addAttribute("result", 0);  return "form2/select1";  }  @PostMapping("form2/select1")  public String select1(Model model, Integer number1, String cmd, Integer number2) {  model.addAttribute("number1", number1);  model.addAttribute("number2", number2);  model.addAttribute("cmd", cmd);  model.addAttribute("result", calc(number1, cmd, number2));  return "form2/select1";  }  private int calc(int number1, String cmd, int number2) {  int result = 0;  switch (cmd) {  case "+": result = number1 + number2; break;  case "-": result = number1 - number2; break;  case "\*": result = number1 \* number2; break;  case "/": result = number1 / number2; break;  }  return result;  }  @GetMapping("form2/select2")  public String select2(Model model) {  model.addAttribute("number1", 0);  model.addAttribute("number2", 0);  model.addAttribute("cmd", "+");  model.addAttribute("result", 0);  return "form2/select2";  }  @PostMapping("form2/select2")  public String select2(Model model, Integer number1, String cmd, Integer number2) {  model.addAttribute("number1", number1);  model.addAttribute("number2", number2);  model.addAttribute("cmd", cmd);  model.addAttribute("result", calc(number1, cmd, number2));  return "form2/select2";  }  } |

select2 액션 메소드 구현은 select1 액션 메소드 구현과 차이가 없다.

리턴하는 뷰의 이름만 다르다.

## form2/select2.jsp 생성

### src/main/webapp/WEB-INF/views/form2/select2.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 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <style>  div.container { width: 400px; margin: 20px auto; }  form div { margin-bottom: 10px; }  label { display: inline-block; width: 70px; text-align: right; }  input { width: 150px; padding: 4px; }  select { width: 50px; padding: 4px; }  button { padding: 0.4em 2em; margin-top: 10px; }  </style>  </head>  <body>  <div class="container">  <form method="post">  <h1>select 02</h1>  <div>  <label>number1:</label>  <input type="text" name="number1" value="${ number1 }" />  </div>    <div>  <label>operator:</label>  <select name="cmd">  <option value="+" ${ cmd == "+" ? "selected" : "" }>+</option>  <option value="-" ${ cmd == "-" ? "selected" : "" }>-</option>  <option value="\*" ${ cmd == "\*" ? "selected" : "" }>\*</option>  <option value="/" ${ cmd == "/" ? "selected" : "" }>/</option>  </select>  </div>  <div>  <label>number2:</label>  <input type="text" name="number2" value="${ number2 }" />  </div>  <div>  <button type="submit">확인</button>  </div>  </form>    <p>  결과: ${ result }  <p>  </div>  </body>  </html> |

|  |
| --- |
| <option value="+" ${ cmd == "+" ? "selected" : "" }>+</option>  <option value="-" ${ cmd == "-" ? "selected" : "" }>-</option>  <option value="\*" ${ cmd == "\*" ? "selected" : "" }>\*</option>  <option value="/" ${ cmd == "/" ? "selected" : "" }>/</option> |

cmd 이름의 model attribute 값이 "-" 이라면, 위 코드의 실행 결과 출력은 다음과 같다.

|  |
| --- |
| <option >+</option>  <option selected>-</option>  <option >\*</option>  <option >/</option> |

# form3/check1 페이지 구현

## Form3Controller.java 생성

### src/main/java/net/skhu/controller/Form3Controller.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 | package net.skhu.controller;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PostMapping;  @Controller  public class Form3Controller {  @GetMapping("form3/check1")  public String check1(Model model) {  return "form3/check1";  }  @PostMapping("form3/check1")  public String check1(Model model, Boolean checkbox1, String radio1, String radio2) {  model.addAttribute("checkbox1", checkbox1);  model.addAttribute("radio1", radio1);  model.addAttribute("radio2", radio2);  return "form3/check1";  }  } |

## form3/check1.jsp 생성

### src/main/webapp/WEB-INF/views/form3/check1.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 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ page import="java.util.Date, java.text.SimpleDateFormat" %>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">  <style>  div.container { width: 400px; margin: 20px auto; }  div.radio label { margin-right: 20px; }  button { padding: 0.4em 2em; margin-top: 10px; }  </style>  </head>  <body>  <div class="container">  <form method="post">  <h3>checkbox</h3>  <div>  <label>  <input type="checkbox" name="checkbox1" value="true" ${ checkbox1 ? "checked" : ""} />  동의하십니까?  </label>  </div>  <hr />  <h3>radio - 성별은?</h3>  <div class="radio">  <label><input type="radio" name="radio1" value="남" ${ radio1 == "남" ? "checked" : "" } /> 남자</label>  <label><input type="radio" name="radio1" value="여" ${ radio1 == "여" ? "checked" : "" } /> 여자</label>  </div>  <hr />  <h3>radio - 혈액형은?</h3>  <div class="radio">  <label><input type="radio" name="radio2" value="A" ${ radio2 == "A" ? "checked" : "" } /> A형</label>  <label><input type="radio" name="radio2" value="B" ${ radio2 == "B" ? "checked" : "" } /> B형</label>  <label><input type="radio" name="radio2" value="AB" ${ radio2 == "AB" ? "checked" : "" } /> AB형</label>  <label><input type="radio" name="radio2" value="O" ${ radio2 == "O" ? "checked" : "" } /> O형</label>  </div>  <div>  <button type="submit">확인</button>  </div>  </form>  </div>  </body>  </html> |

<input type="checkbox" name="checkbox1" value="true" />

체크박스 태그이다.

이 체크박스가 체크된 상태에서 입력폼의 submit 버튼이 클릭되면,

request parameter에 name="checkbox1" value="true" 값이 포함된다.

체크되지 않은 상태에서 입력폼의 submit 버튼이 클릭되면, request parameter에 데이터가 포함되지 않는다.

<input type="checkbox" name="checkbox1" value="true" checked />

체크박스가 처음 화면에 그려질 때, 체크된 상태로 그려진다.

<input type="radio" name="radio1" value="남" />

<input type="radio" name="radio1" value="여" />

radio 버튼 태그이다.

name 값이 같은 radio 버튼들 중에서 한 개만 체크될 수 있고, 동시에 여러 개 체크 될 수 없다.

value="여" 버튼이 체크된 상태에서 입력폼의 submit 버튼이 클릭되면,

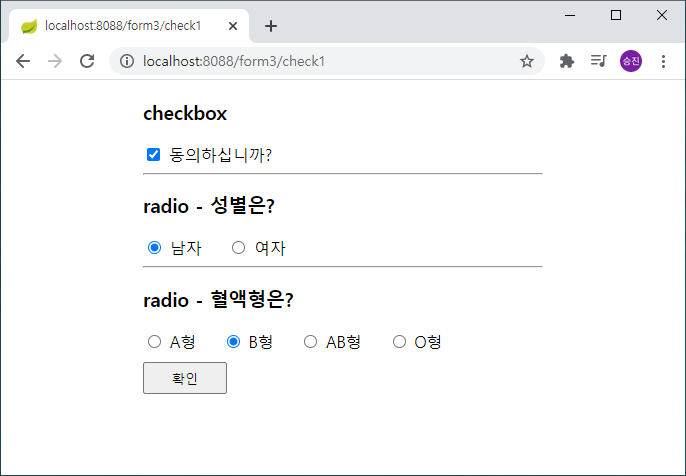
request parameter에 name="radio1" value="여" 값이 포함된다.

<input type="radio" name="radio1" value="남" checked />

radio 버튼이 처음 화면에 그려질 때, 체크된 상태로 그려진다.

## 실행

http://localhost:8088/form3/check1

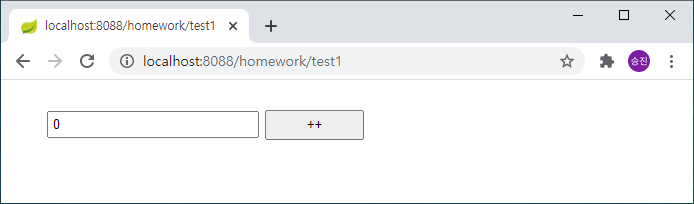


확인 버튼을 누른 후에도 체크 상태가 유지된다.

실행 과정을 분석해 보라.

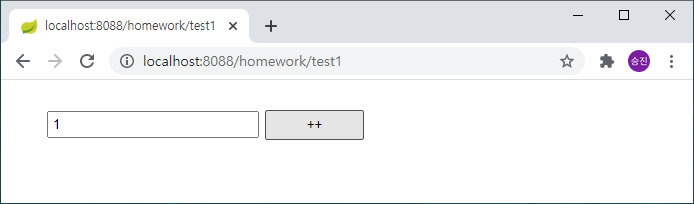
# 연습문제

### homework/test1



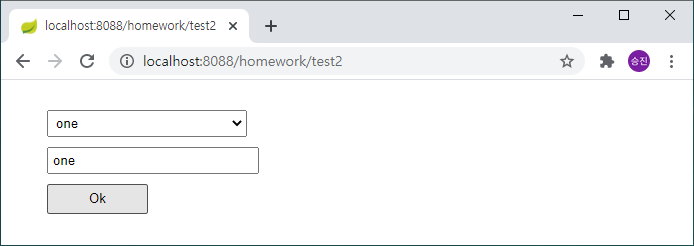
처음 실행하면 위와 같이 출력된다.

++ 버튼을 클릭하면 다음과 같이 출력된다.



++ 버튼을 클릭할 때 마다, 텍스트 박스에 채워진 숫자가 1씩 증가해야 한다.

### homework/test2

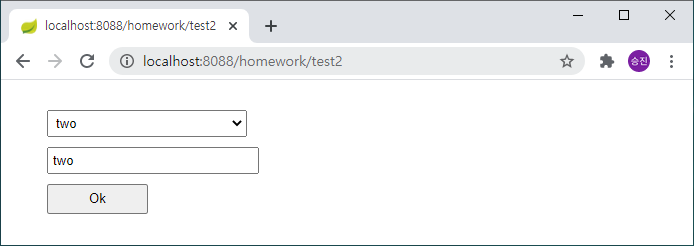


처음 실행하면 위와 같이 출력된다.

select 태그에, one, two, three 세 항목이 들어있다.

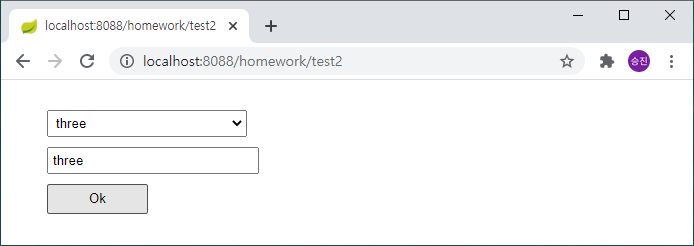
select 태그에서 two를 선택하고 Ok 버튼을 클릭하면,

다음과 같이 출력된다.



select 태그에서 three를 선택하고 Ok 버튼을 클릭하면,

다음과 같이 출력된다.



힌트:

<input type="text"> 태그에는 name 속성을 정의하지 않고, value 속성만 정의해야 한다.

구현 연습만 하면 되고, 제출할 필요 없다.