**Spring MVC 기초**

2021-09-01

이승진

**학습목표**

Spring Boot 프로젝트를 생성한다.

Spring Controller와 View를 구현한다.

Spring Web MVC 실행 절차를 이해한다.

**목차**

[1. 프로젝트 닫기 2](#_Toc81457653)

[2. hello2 프로젝트 3](#_Toc81457654)

[1) 목표 3](#_Toc81457655)

[2) 프로젝트 생성 3](#_Toc81457656)

[3) pom.xml 파일 수정 5](#_Toc81457657)

[4) application.properties 수정 6](#_Toc81457658)

[3. FirstController 7](#_Toc81457659)

[1) Product.java 생성 7](#_Toc81457660)

[2) FirstController.java 생성 9](#_Toc81457661)

[3) 실행 11](#_Toc81457662)

[4) FirstController 액션 메소드 실행 절차 13](#_Toc81457663)

[5) JSON 형태의 문자열 13](#_Toc81457664)

[4. SecondController 14](#_Toc81457665)

[1) 뷰(view) 14](#_Toc81457666)

[2) 실행 순서 14](#_Toc81457667)

[3) 뷰 파일 위치 설정 15](#_Toc81457668)

[4) 뷰 폴더 생성 16](#_Toc81457669)

[5) second/test1.jsp 뷰 파일 생성 17](#_Toc81457670)

[6) second/test2.jsp 뷰 파일 생성 19](#_Toc81457671)

[7) SecondController.java 생성 20](#_Toc81457672)

[8) 실행 23](#_Toc81457673)

[9) 실행 절차 26](#_Toc81457674)

[5. HomeController 27](#_Toc81457675)

[1) HomeController.java 생성 27](#_Toc81457676)

[2) home/index.jsp 생성 28](#_Toc81457677)

[3) 실행 29](#_Toc81457678)

[6. 과제 30](#_Toc81457679)

[1) 프로젝트 생성 30](#_Toc81457680)

[2) Student 클래스 생성 30](#_Toc81457681)

[3) ThirdController 클래스 생성 32](#_Toc81457682)

[4) third/test1.jsp 생성 32](#_Toc81457683)

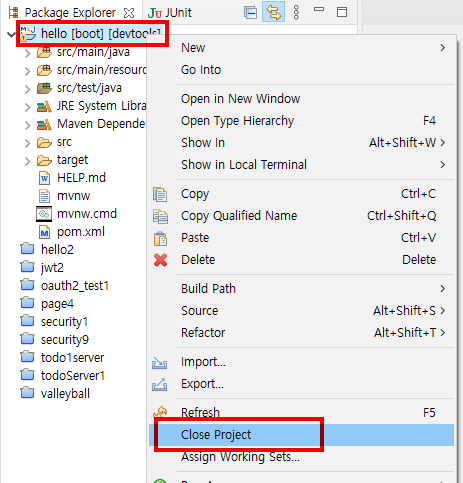
[5) URL 32](#_Toc81457684)

# 프로젝트 닫기

현재 중인 프로젝트만 열어 놓고, 나머지 프로젝트들은 닫으면,

사소한 차이이지만 eclipse가 좀 더 빠르게 실행된다.

### hello 프로젝트 닫기



닫은 프로젝트를 다시 열려면, 프로젝트 이름을 더블 클릭하자.

# hello2 프로젝트

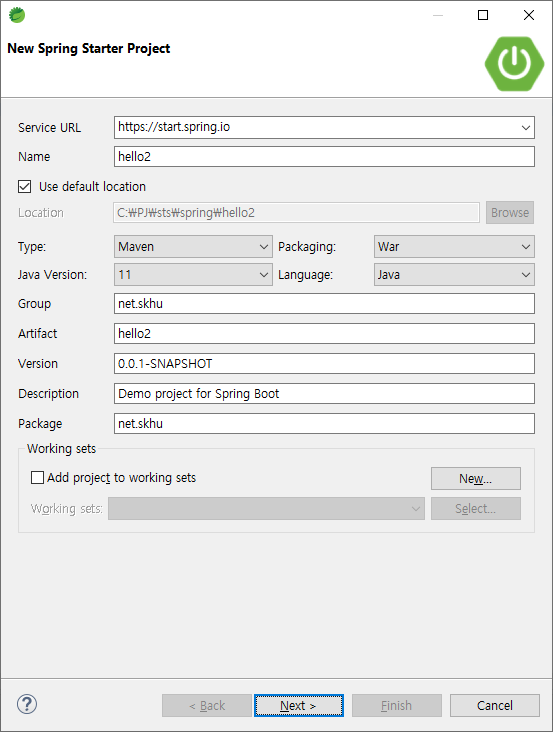
## 목표

간단한 Spring Web MVC 프로젝트를 만들고 실행해 보자.

Spring Boot 기술을 이용하면 Spring MVC 프로젝트를 아주 간단하게 시작할 수 있다.

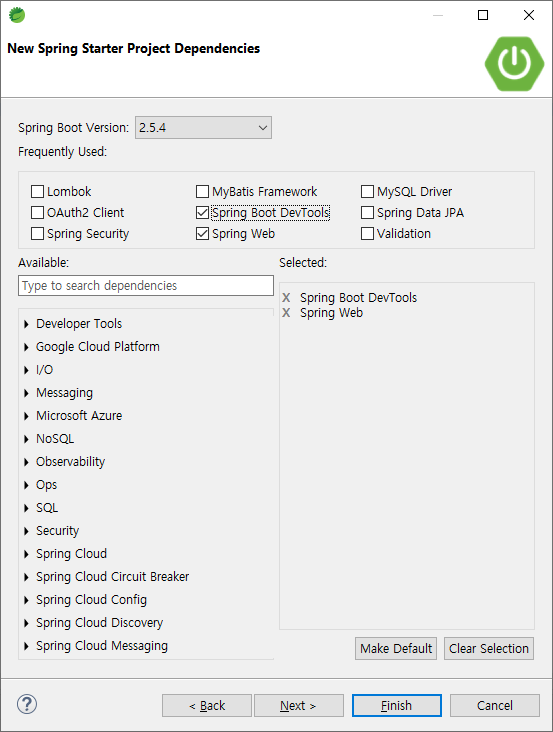
## 프로젝트 생성

메뉴: File - New Spring Starter Propject



|  |  |
| --- | --- |
| Name | hello2 |
| Type | maven |
| Packaging | war |
| Group | net.skhu |
| Artifact | hello |
| Package | net.skhu |

Spring Boot 기술을 이용한 프로젝트 생성 시작 화면이다.



Spring Boot Version: 가급적 위 버전과 가장 가까운 2.x.x 버전을 선택하자.

spring boot가 계속 버전 업이 되면서, 호환성 문제가 발생하기도 한다.

Spring Boot DevTools, Spring Web 체크

이 예제에서는 데이터베이스 연결을 하지 않을 것이기 때문에,

데이터베이스 항목들을 체크하지 않았다.

### Spring Web 항목:

Web 항목은 Spring Web MVC 라이브러리의 사용 여부를 설정하는 것다.

Spring Web MVC 개발을 하려고 하므로 필수 항목이다.

Spring Web MVC

### Spring Boot DevTools 항목:

프로젝트를 실행할 때, 프로젝트를 재시작하지 않아도 수정된 소스 코드가 실행에 즉시 반영될 수 있도록 해주는 개발 도구

## pom.xml 파일 수정

|  |
| --- |
| <dependency>  <groupId>jstl</groupId>  <artifactId>jstl</artifactId>  <version>1.2</version>  </dependency>  <dependency>  <groupId>org.apache.tomcat.embed</groupId>  <artifactId>tomcat-embed-jasper</artifactId>  </dependency> |

pom.xml 파일을 열고 dependencies 태그 내부에 위 내용을 추가하자.

pom.xml 파일은 maven의 설정 파일이다.

위 dependency 항목들은 다음과 같은 라이브러리를 프로젝트에 포함하기 위함이다.

위 설정에 의해서 maven이 이 라이브러리를 자동으로 다운로드해서 프로젝트에 설치해 준다.

### JSTL

JSTL 확장 태그를 사용하기 위해 필요한 라이브러리

<c:forEach> <c:if>

### tomcat-embed-jasper

스프링 부트 앱에는 톰캣 서버가 내장되어 있다.

톰캣 서버는 servlet container 이다.

톰캣 서버는 servlet을 실행하는 서버이다.

톰캣에서 JSP 파일이 실행되는 것은, 먼저 JSP 파일이 servlet이로 변환(컴파일)된 후 실행된다.

tomcat-embed-jasper는 JSP 파일을 servlet으로 변환하는 컴파일러이다.

스프링 부트 앱에 내장된 톰캣 서버에는 tomcat-embed-jasper가 내장되어 있지 않다.

이 것을 프로젝트에 추가하지 않고 실행하면,

JSP 파일이 실행되지 않고, 웹브라우저에서 다운로드 될 것이다.

톰캣은 servlet만 실행할뿐, 다른 파일들은 웹브라우저로 그대로 전송하기 때문이다.

## application.properties 수정

src/main/resources/application.properties

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

스프링 부트의 설정 파일

뷰 파일의 위치에 대한 설정만 포함한다.

spring.mvc.view.prefix=/WEB-INF/views/ -> 뷰 파일 위치

spring.mvc.view.suffix=.jsp -> 뷰 파일의 확장자

내장된 톰캣 서버가 실행될 포트 디폴트 포트는 8080인데,

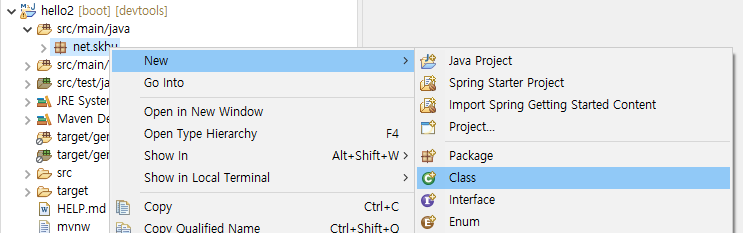
8080 포트를 사용하는 다른 서버와 충돌이 발생할 확률이 높다.

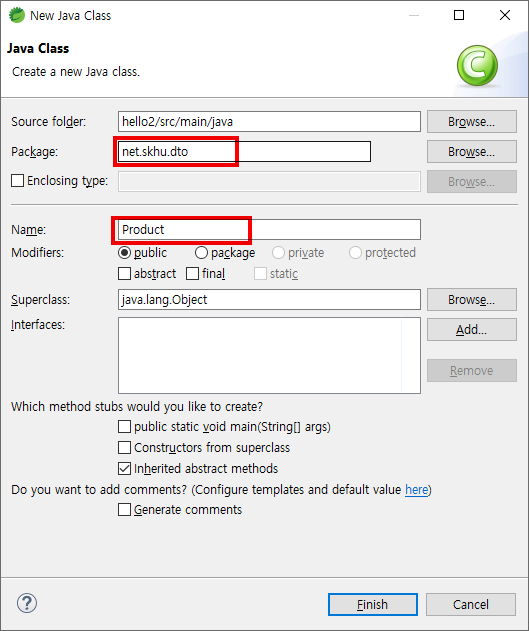
8088 포트로 변경하자.

server.port=8088 -> 톰캣 서버가 사용할 포트 번호

# FirstController

## Product.java 생성





|  |  |
| --- | --- |
| Package | net.skhu.dto |
| Name | Product |

### src/main/java/net/skhu/dto/Product.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 | package net.skhu.dto;  public class Product {  String name;  int unitCost;  public Product(String name, int unitCost) {  this.name = name;  this.unitCost = unitCost;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public int getUnitCost() {  return unitCost;  }  public void setUnitCost(int unitCost) {  this.unitCost = unitCost;  }  } |

제품(product)의 이름(name)과 가격(cost) 데이터를 넣어서 전달하기 위한 클래스이다.

전달 => 파라미터로 전달, 리턴값으로 전달

이런 목적의 클래스를 DTO(Data Transfer Object) 라고 부른다.

## FirstController.java 생성

### src/main/java/net/skhu/controller/FirstController.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 | package net.skhu.controller;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RestController;  import net.skhu.dto.Product;  @RestController  @RequestMapping("first")  public class FirstController {  @RequestMapping("test1")  public String test1() {  return "안녕하세요";  }  @RequestMapping("test2")  public String[] test2() {  return new String[] { "월", "화", "수", "목", "금", "토", "일" };  }  @RequestMapping("test3")  public Product test3() {  return new Product("맥주", 2000);  }  @RequestMapping("test4")  public Product[] test4() {  return new Product[] {  new Product("맥주", 2000),  new Product("우유", 1500)  };  }  } |

### 컨트롤러 클래스

컨트롤러 클래스는 웹브라우저의 URL 요청을 받아서, 웹서버에서 실행되는 클래스이다.

웹브라우저가 웹서버에 어떤 URL을 요청을 하면, 그 URL에 해당하는 컨트롤러의 액션 메소드가 자동으로 호출되어 실행된다. 컨트롤러 클래스는 @RestController, @Controller 어노테이션 중의 하나가 붙어있어야 한다.

### 액션 메소드

웹브라우저가 어떤 URL을 웹서버에 요청하면, 그 요청된 URL에 해당하는 컨트롤러의 어떤 메소드가 자동으로 호출된다. 이렇게 웹브라우저의 요청에 의해서 자동으로 호출되는 컨트롤러의 메소드를 액션 메소드라고 부른다.

### @RequestMapping 어노테이션과 액션 메소드 URL

컨트롤러 클래스에 붙은 @RequestMapping("**first**") 어노테이션의 **first** 부분과

액션 메소드에 붙은 @RequestMapping("**test1**") 어노테이션의 **test** 부분이 결합된 **first/test1** 이

그 액션 메소드의 URL 이다.

Run as Spring Boot App 메뉴로 실행할 경우,

**http://localhost:8088/first/test1** URL을 웹브라우저가 웹서버에 요청하면,

웹서버에서 FirstController 컨트롤러 클래스의 test1 액션 메소드가 자동으로 호출되어 실행된다.

자동으로 호출할 액션 메소드를 찾을 때 컨트롤러 클래스 이름이나 액션 메소드의 이름은 상관 없고,

@RequestMapping 어노테이션에 등록된 URL만 일치하면 된다.

### @RestController

이 어노테이션이 붙은 컨트롤러를 레스트 컨트롤러라고 부르자.

레스트 컨트롤러는, 보통의 컨트롤러와 조금 다르다.

그 차이점은 액션 메소드의 리턴값이 무엇이냐는 점이다.

레스트 컨트롤러의 액션 메소드의 리턴값은 데이터이다.

그 데이터는 웹브라우저에 전송된다.

레스트 컨트롤러의 액션 메소드가 리턴하는 것이 String 문자열이면, 그 문자열이 웹브라우저에 그대로 전송된다.

레스트 컨트롤러의 액션 메소드가 리턴하는 것이 배열이나 Java 객체이면, JSON 형태로 변환되어 웹브라우저에 전송된다.

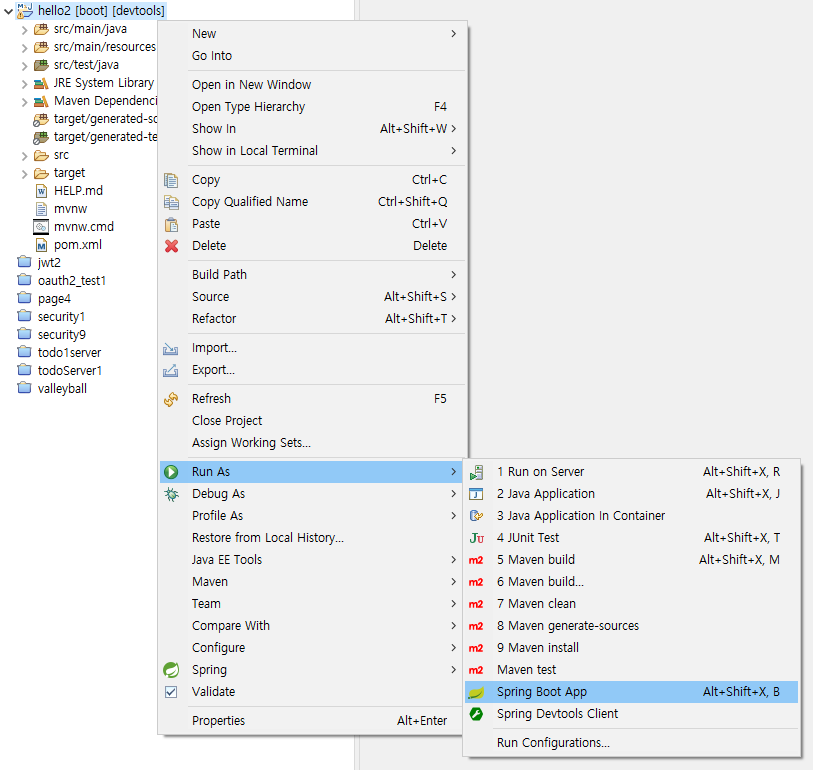
### @Controller

보통의 컨트롤러의 액션메소드의 리턴값은 뷰(view)의 이름이다.

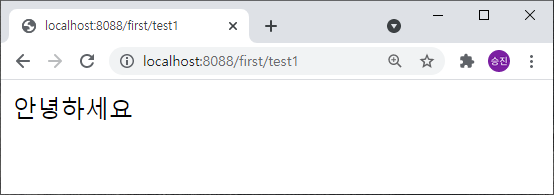
액션메소드가 리턴한 후, 그 이름의 뷰가 실행된다.

뷰의 실행결과 출력이 웹브라우저에 전송된다.

## 실행



### http://localhost:8088/first/test1



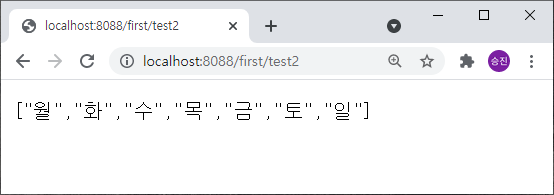
test1 액션 메소드가 리턴한 문자열이 그대로 웹브라우저에 전송되었다.

internet explorer 웹브라우저에서 실행하지 말고,

다른 웹브라우저에서 실행하자. (chrome, firefox, safari)

internet explorer 웹브라우저는 조만간 단종된다.

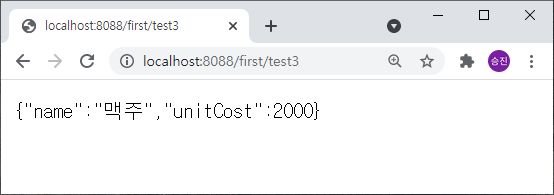
### http://localhost:8088/first/test2



test2 액션 메소드가 리턴한 String 배열이 JSON 포멧으로 웹브라우저에 전송되었다.

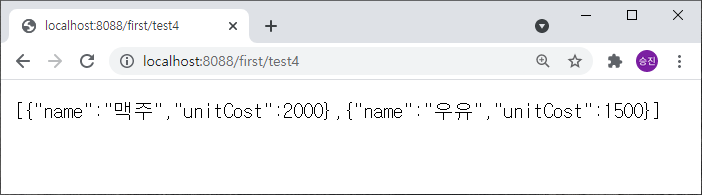
웹브라우저에 보이는 내용이 Stirng 배열 JSON 포맷이다.

### http://localhost:8088/first/test3



test3 액션 메소드가 리턴한 Product 객체가 JSON 포멧으로 웹브라우저에 전송되었다.

웹브라우저에 보이는 내용이 객체 한 개의 JSON 포맷이다.

**http://localhost:8088/first/test4** 

test4 액션 메소드가 리턴한 Product 객체 배열이 JSON 포멧으로 웹브라우저에 전송되었다.

웹브라우저에 보이는 내용이 객체 배열의 JSON 포맷이다.

## FirstController 액션 메소드 실행 절차



|  |  |
| --- | --- |
| 1 | 웹브라우저가 서버에 **요청(HTTP Request)**을 전달한다.  이 요청에는, 요청 대상을 가르키는 **URL**이 담겨 있다.  (URL: http://localhost:8088/first/test3) |
| 2 | 웹브라우저로부터 서버에 전달된 요청을 Spring Web MVC 엔진이 받는다.  스프링 엔진은 요청된 URL과 일치하는 **컨트롤러 액션 메소드**를 찾아서 호출한다.  (FirstController 클래스의 test3() 액션 메소드 호출) |
| 3 | FirstController의 test3() 액션 메소드는 Java 객체를 리턴한다.  (Product 객체 리턴) |
| 4 | 리턴된 Java 객체는, Spring Web MVC 엔진에 의해서 JSON 형태의 문자열로 변경된다.  {"name":"맥주","unitCost":2000, "quantity":120} |
| 5 | JSON 형태의 문자열이 웹브라우저에 전송된다.  이 전송은 최초 웹브라우저의 요청(http request)에 대한 응답(http response)이다. |
| 6 | 웹 서버로부터 전송된 JSON 형태의 문자열이 웹브라우저에 표시된다. |

## JSON 형태의 문자열

서버와 데이터를 주고 받을 때, 데이터를 JSON 형태의 문자열로 변환해서 전송하는 경우가 흔하다.

JSON 숫자 배열

|  |
| --- |
| [1, 2, 3, 4, 5] |

JSON 문자열 배열

|  |
| --- |
| ["월","화","수","목","금","토","일"] |

JSON 객체 예#1

|  |
| --- |
| {"name":"맥주","unitCost":2000, "quantity":120} |

JSON 객체 예#2

|  |
| --- |
| {"studentNumber":"201532045","name":"홍길동", "email":"hgd@skhu.ac.kr" } |

JSON 객체 배열의 예#1

|  |
| --- |
| [{"name":"맥주","unitCost":2000},{"name":"우유","unitCost":1500}] |

JSON 객체 배열의 예#2

|  |
| --- |
| [{"studentNumber":"201532045","name":"홍길동", "email":"hgd@skhu.ac.kr" },  {"studentNumber":"201532046","name":"전우치", "email":"jwc@skhu.ac.kr" },  {"studentNumber":"201532047","name":"임꺽정", "email":"ikj@skhu.ac.kr" }] |

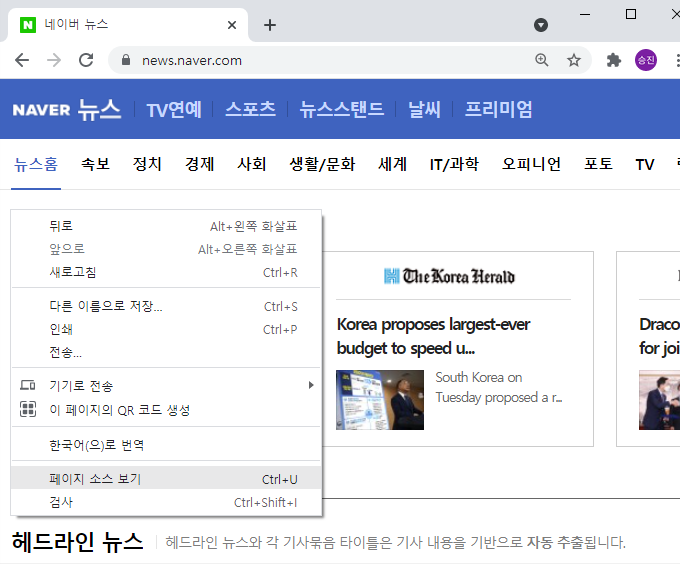
# SecondController

## 뷰(view)

웹브라우저의 요청(http request)에 대한 응답(http response)으로,

웹서버에서 웹브라우저로 전송되는 것은 대부분 html 태그이다.

우리가 웹서핑하면서 보는 웹페이지들의 내용이 html 태그이다.



웹페이지를 마우스 오른쪽 버튼으로 클릭하고, "페이지 소스 보기" 메뉴를 클릭하면

웹서버에서 웹브라우저로 전송된 내용인, html 태그 소스를 볼 수 있다.

컨트롤러 액션 메소드 실행 결과로 웹브라우저로 전송되는 것이 html 태그인 경우에는

뷰(view) 파일을 구현해야 한다.

## 실행 순서

웹브라우저에서 웹서버에 요청(http request)이 전달되면,

요청된 URL과 일치하는, 컨트롤러의 액션 메소드가 실행된다.

그리고 액션 메소드의 뒤를 이어서 뷰(view) 파일이 실행된다.

뷰 파일의 실행 결과 출력된 html 태그들이 웹브라우저로 전송된다.

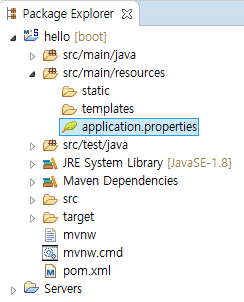
## 뷰 파일 위치 설정

Spring Boot 기술을 이용하여 생성된 Spring MVC 프로젝트의 설정 파일은

**src/main/resources/application.properties** 파일이다.

이 파일에 뷰(view) 파일들의 위치를 지정해야 한다.

### application.properties 파일 수정



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

|  |  |
| --- | --- |
| 1  2  3 | spring.mvc.view.prefix=/WEB-INF/views/  spring.mvc.view.suffix=.jsp  server.port=8088 |

이 파일의 줄 끝에 공백이 있으면 안된다. 주의하자.

뷰 파일들은 **src/main/webapp/WEB-INF/views** 폴더 아래에 \*.jsp 파일이어야 한다.

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

뷰 파일들이 위치할 폴더를 지정한다.

|  |
| --- |
| spring.mvc.view.suffix=.jsp |

뷰 파일들의 확장자를 지정한다.

예를 들어 액션 메소드가 리턴한 뷰의 이름이 다음과 같으면

|  |
| --- |
| return "second/test1"; |

이 뷰 파일의 위치는 **/WEB-INF/views/second/test1.jsp** 이어야 한다.

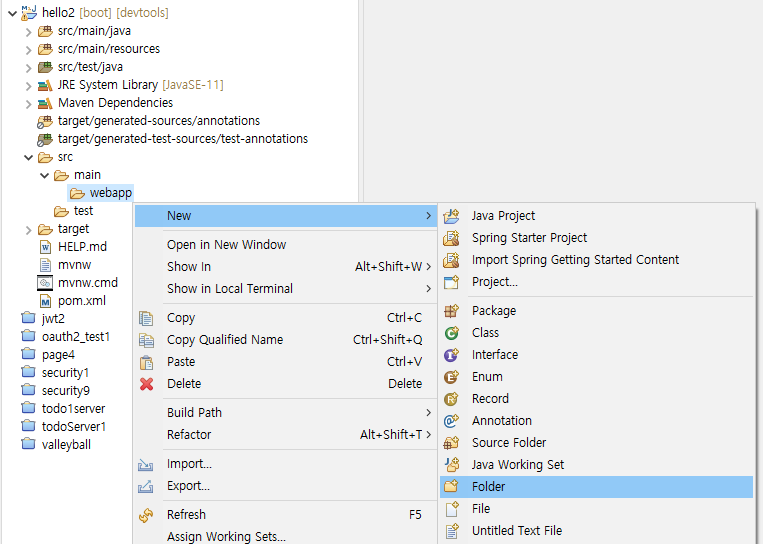
만약 폴더 명이나 파일명에 오타가 있다면, 뷰 파일을 찾지 못할 테니 에러가 발생할 것이다.

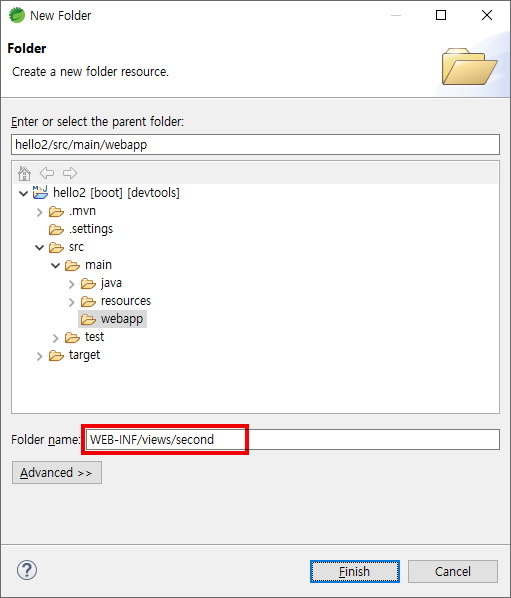
이 경우 웹브라우저에 "404 Not Found" 메시지가 보일 것이다.

**/WEB-INF/views/** 폴더 아래에, 컨트롤러 클래스 이름으로 폴더를 만들고, 그 폴더에 액션 메소드 이름으로 뷰 파일들을 만드는 것이 관례이다. (꼭 그렇게 해야만 하는 것은 아니다)

## 뷰 폴더 생성

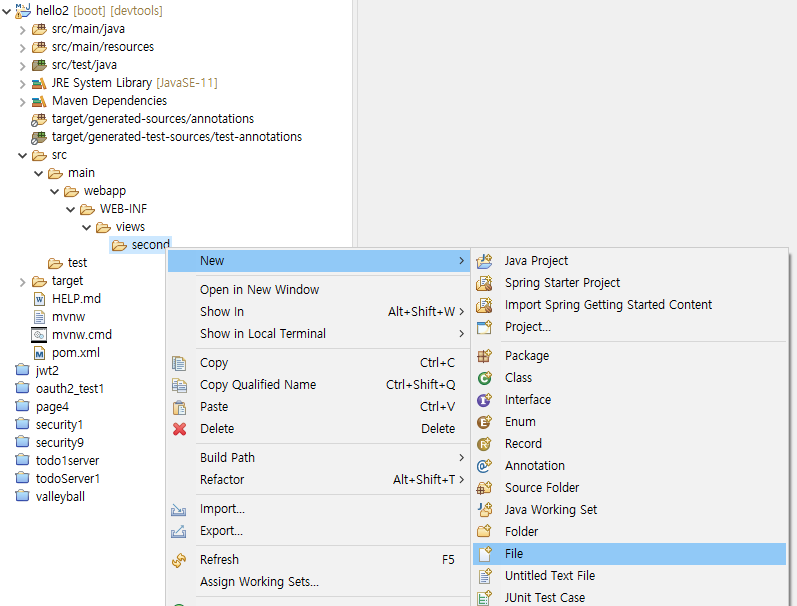
### webapp/WEF-INF/views/second 폴더 생성

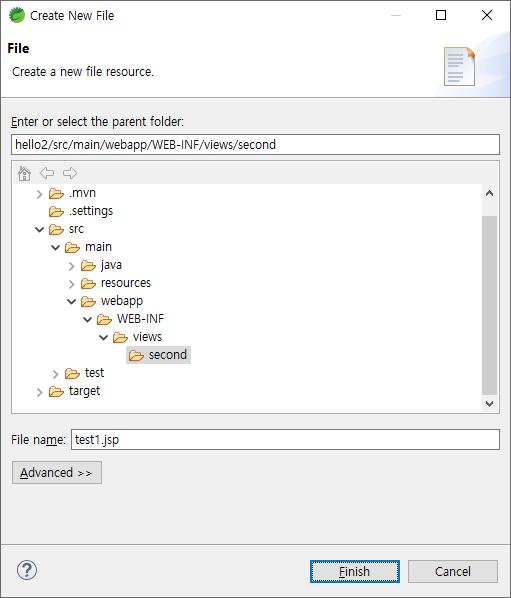




## second/test1.jsp 뷰 파일 생성

src/main/webapp/WEB-INF/views/second 폴더 아래에 test1.jsp 파일을 생성하자.





### src/main/webapp/WEB-INF/views/second/test1.jsp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <%@ 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">  </head>  <body>  <h1>${ message }</h1>  </body>  </html> |

(줄1)

|  |
| --- |
| <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%> |

JSP 파일의 첫 줄은 언제나 위와 같아야 한다. 그래야 한글이 깨지지 않는다.

그리고 개발 도구의 JSP 파일의 인코딩도 "UTF-8" 이어야 한다. (spring tool suite 설치 강의노트 참고)

(줄8)

|  |
| --- |
| ${ message } |

test1.jsp 뷰 파일에 전달된 데이터 중에서, 이름이 "message"인 데이터(model attribute)를 여기에 출력하라는 명령이다.

test1.jsp 파일의 내용 중에서 아래, (줄1) 전체와 (줄8) ${ message } 부분을 제외한 나머지는 다음과 같다.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  </head>  <body>  <h1> </h1>  </body>  </html> |

위의 내용을 평범한 HTML 태그들이다.

JSP 파일의 내용 중에서 평범한 HTML 태그들은 웹브라우저에 그대로 출력된다.

(줄1) 부분은 웹브라우저에 출력되지 않는다.

(줄8)의 ${ message } 부분은, model attribute 값만 출력된다.

## second/test2.jsp 뷰 파일 생성

### src/main/webapp/WEB-INF/views/second/test2.jsp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <%@ 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">  </head>  <body>  <h1>${ product.name }</h1>  <h2>${ product.unitCost }</h2>  </body>  </html> |

|  |
| --- |
| ${ product.name } |

test2.jsp 뷰 파일에 전달된 데이터(model attribute) 중에서,

이름이 "product"인 객체의 getName() 메소드를 호출하고,

이 getName() 메소드가 리턴한 값을, 여기에 출력하라는 명령이다.

객체의 name 멤버 변수 값을 출력하는 것이 아니고, getName() 메소드의 리턴값을 출력한다.

|  |
| --- |
| ${ product.unitCost } |

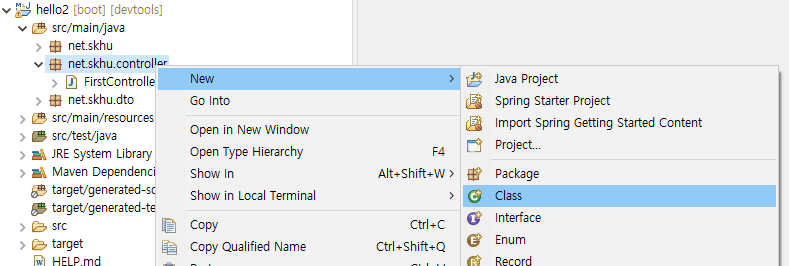
test2.jsp 뷰 파일에 전달된 데이터 중에서,

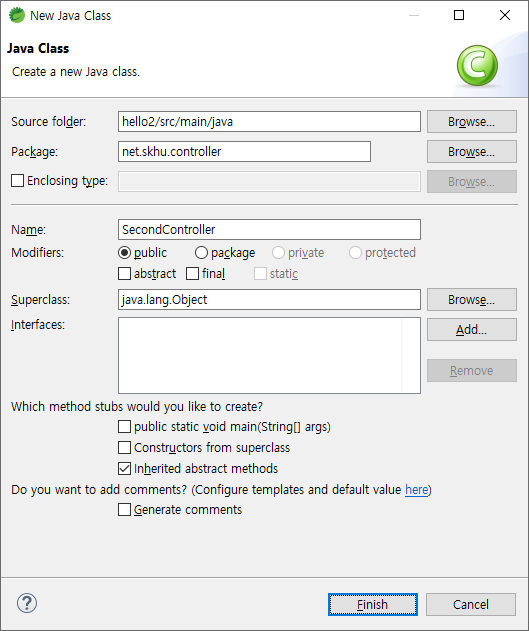
이름이 "product"인 객체의 getUnitCost() 메소드를 호출하고,

이 getUnitCost() 메소드가 리턴한 값을, 여기에 출력하라는 명령이다.

객체의 unitCost 멤버 변수 값을 출력하는 것이 아니고, getUnitCost() 메소드의 리턴값을 출력한다.

## SecondController.java 생성





### src/main/java/net/skhu/SecondController.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 | package net.skhu.controller;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.RequestMapping;  import net.skhu.dto.Product;  @Controller  @RequestMapping("second")  public class SecondController {  @RequestMapping("test1")  public String test1(Model model) {  model.addAttribute("message", "안녕하세요");  return "second/test1";  }  @RequestMapping("test2")  public String test2(Model model) {  Product product = new Product("맥주", 2000);  model.addAttribute("product", product);  return "second/test2";  }  } |

### @Controller

컨트롤러 클래스에 이 어노테이션(annotation)을 붙여주어야 한다.

이것을 붙여주지 않으면 컨트롤러 클래스가 실행되지 않는다. (404 Not Found 에러)

이 컨트롤러의 액션 메소드는 문자열을 리턴한다. 이 문자열은, 뷰 파일의 이름이다.

이 컨트롤러의 액션 메소드가 리턴한 후, 리턴된 이름의 뷰 파일이 실행된다.

### Model 객체

|  |
| --- |
| public String test1(Model model) { |

컨트롤러의 액션 메소드는 Spring Web MVC 엔진에 의해서 호출된다.

대부분의 액션 메소드의 파라미터에 Model model 객체가 포함된다.

컨트롤러의 액션 메소드가 뷰 파일에 전달할 데이터를, Model 객체에 넣어서 전달한다.

즉 Model 객체는, 데이터 전달 상자라고 보면 된다.

이렇게 전달되는 데이터를 model attribute 라고 한다.

|  |
| --- |
| model.addAttribute("message", "안녕하세요"); |

뷰 파일에 전달할 데이터를 Model 객체에 넣는 코드이다.

전달되는 데이터의 이름은 "message" 이고, 값은 "안녕하세요" 문자열이다.

이렇게 전달된 문자열을 뷰 파일에서 출력하는 코드는 아래와 같다.

|  |
| --- |
| ${ message } |

|  |
| --- |
| Product product = new Product("맥주", 2000);  model.addAttribute("product", product); |

뷰 파일에 전달할 객체를 Model 객체에 넣는 코드이다.

전달되는 데이터의 이름은 "product" 이고, 값은 Product 객체이다.

이렇게 전달된 문자열을 뷰 파일에서 출력하는 코드는 아래와 같다.

|  |
| --- |
| ${ product.name } |

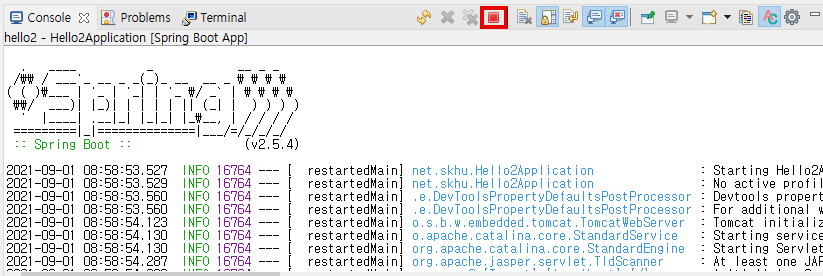
getName() 메소드의 리턴값이 출력된다.

|  |
| --- |
| ${ product.unitCost } |

getUnitCost() 메소드의 리턴값이 출력된다.

## 실행

### 주의사항



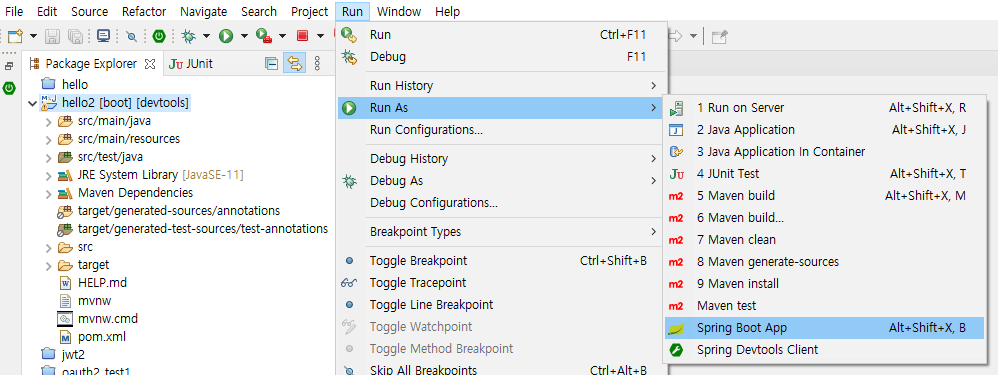
만약 console 창의 내용이 위와 같다면, 이미 프로젝트가 8088 포트에서 실행 중이다.

(빨간색 버튼에 주목하자. 이 버튼을 클릭하면 프로젝트 실행이 종료된다)

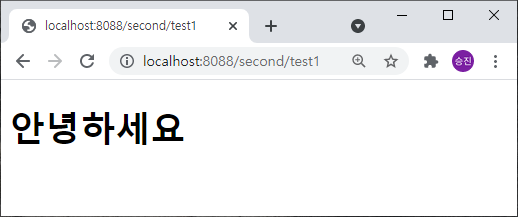
이 상태에서는 Run 메뉴를 또 실행하면 안된다.

프로젝트를 또 실행되어 8088 포트 충돌이 발생할 것이다.

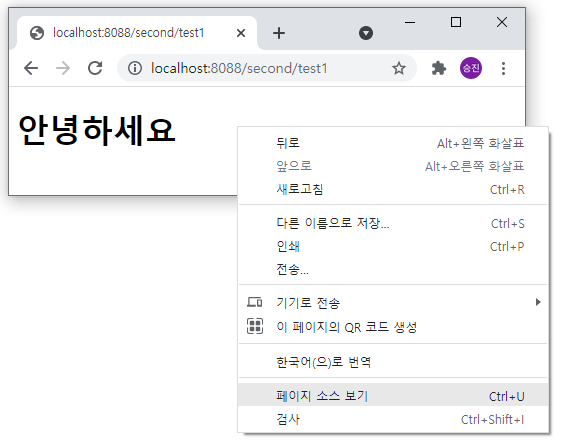
### 실행



### http://localhost:8088/second/test1



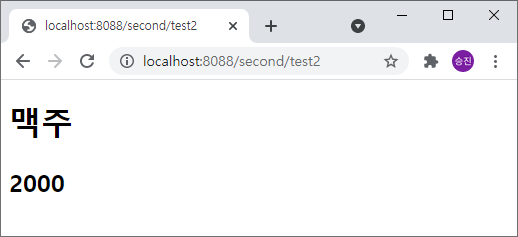
### second/test1 실행화면 소스보기



|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  </head>  <body>  <h1>안녕하세요</h1>  </body>  </html> |

웹브라우저로 전달된 (http response) html 태그는 위와 같다.

### http://localhost:8088/second/test2



|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  </head>  <body>  <h1>맥주</h1>  <h2>2000</h2>  </body>  </html> |

웹브라우저로 전달된 (http response) html 태그는 위와 같다.

## 실행 절차



|  |  |
| --- | --- |
| 1 | 웹브라우저가 웹서버에 **요청(HTTP Request)**을 전달한다.  이 요청에는, 요청 대상을 가르키는 **URL**이 담겨 있다. |
| 2 | 웹브라우저로부터 웹서버에 전달된 요청을 Spring Web MVC 엔진이 받는다.  스프링 엔진은 요청된 URL과 일치하는 **컨트롤러 액션 메소드**를 찾아서 호출한다. |
| 3 | 컨트롤러의 액션 메소드는 데이터를 Model 객체에 넣는다. |
| 4 | 컨트롤러의 액션 메소드는 뷰의 이름을 리턴한다. |
| 5 | 그 이름의 뷰가 실행된다. |
| 6 | 뷰는 Model 객체에서 데이터를 꺼내서 출력하고, HTML 태그들도 출력한다. |
| 7 | 뷰가 출력한 HTML 태그들이 웹브라우저에 전송된다.  이 전송은 최초 웹브라우저의 요청(http request)에 대한 응답(http response)이다. |
| 8 | 웹 서버로부터 전송된 HTML 태그들이 웹브라우저에 표시된다. |

# HomeController

## HomeController.java 생성

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | package net.skhu.controller;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.RequestMapping;  @Controller  public class HomeController {  @RequestMapping("/")  public String index() {  return "home/index";  }  } |

### 액션 메소드

@RequestMapping("**/**") 어노테이션의 "**/**" 부분은

그 액션 메소드를 호출하기 위한 URL 이다.

### context path

액션 메소드의 URL 앞에 **http://서버주소/contextPath** 을 붙인 URL이, 그 액션 메소드를 호출하기 위한 URL 이다.

예를 들어 context path가 "**/**" 이라면, http://localhost:8088**/** URL이 요청되면, 이 액션 메소드가 호출된다.

context path가 "**/hello2**" 이라면, http://localhost:8088**/hello2** URL이 요청되면, 이 액션 메소드가 호출된다.

Run - Run As - Spring Boot App 메뉴로 실행한 경우 (Spring Boot App에 내장된 tomcat에서 실행한 경우)

context path는 "/" 이다.

Run - Run on Server 메뉴로 실행한 경우 (따로 설치한 tomcat 서버에서 실행한 경우)

context path는 "/프로젝트이름" 이다.

따로 설치한 tomcat 서버가 없다면, Run - Ron on Server 메뉴로 실행할 수 없다.

### "home/index"

index 메소드가 리턴한 문자열은 "home/index" 이다.

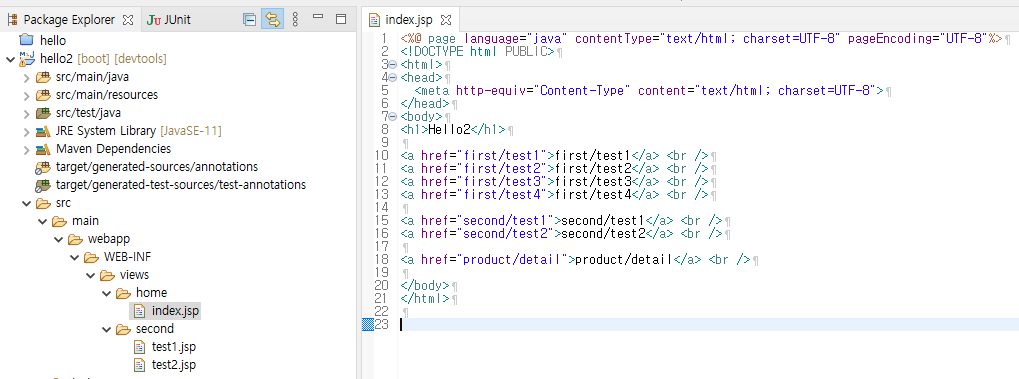
/WEB-INF/views/home/index.jsp

이 뷰 파일이 없으면 실행할 때 404 Not Found 에러가 발생한다.

## home/index.jsp 생성

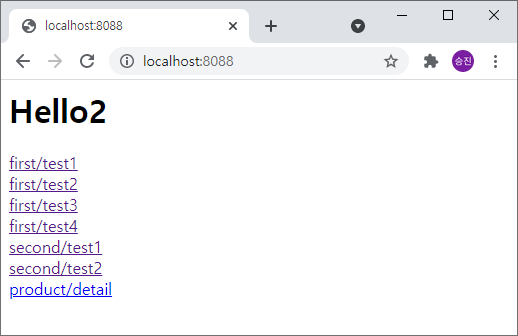
src/main/webapp/WEB-INF/views/home/index.jsp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <!DOCTYPE html PUBLIC>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  </head>  <body>  <h1>Hello2</h1>  <a href="first/test1">first/test1</a> <br />  <a href="first/test2">first/test2</a> <br />  <a href="first/test3">first/test3</a> <br />  <a href="first/test4">first/test4</a> <br />  <a href="second/test1">second/test1</a> <br />  <a href="second/test2">second/test2</a> <br />  <a href="product/detail">product/detail</a> <br />  </body>  </html> |

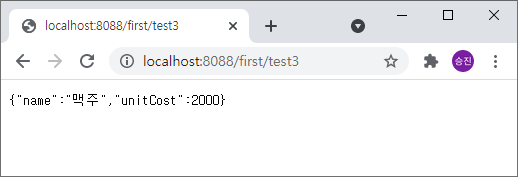


## 실행

### (1) http://localhost:8088



### (2) first/test3 클릭



### 위 실행 절차

|  |  |
| --- | --- |
| 1 | hello2 프로젝트를 실행하면, hello2 프로젝트가 빌드되고,  빌드된 배포 파일(hello2.war) 내부의 톰켓 서버가 실행된다. |
| 2 | 웹브라우저가 **http://localhost:8088/** URL을 요청하면,  서버에서 HomeController의 index 메소드가 호출된다. |
| 3 | /WEB-INF/views/home/index.jsp 파일이 실행되고,  그 실행결과 출력이 웹브라우저에 전송된다. |
| 4 | 전송된 html 태그가 웹브라우저에 표시된다. |
| 5 | 사용자가 웹브라우저에서 first/test3 링크를 마우스로 클릭한다. |
| 6 | 웹브라우저가 **http://locahost:8088/first/test3** URL을 웹서버에 요청한다. |
| 7 | FirstController의 test3 액션 메소드가 서버에서 실행된다. |
| 8 | test3 액션 메소드가 리턴한 데이터가 웹브라우저에 전송된다. |
| 9 | 전송된 데이터가 웹브라우저에 표시된다. |

# 과제

## 프로젝트 생성

프로젝트명: hwA학번

예: hwA200814199

## Student 클래스 생성

### 멤버 변수

int id;

String studentNumber;

String studentName;

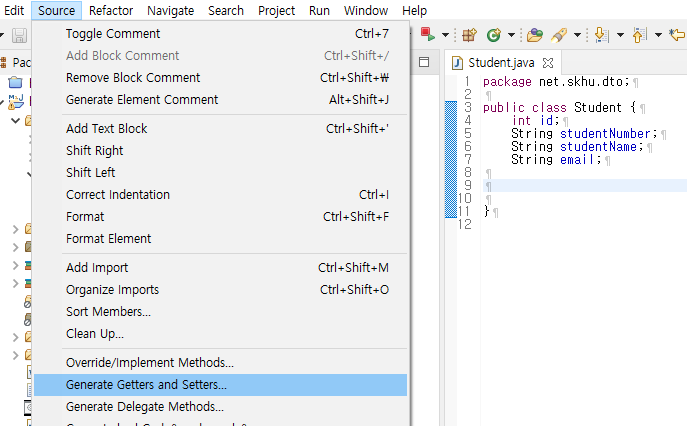
String email;

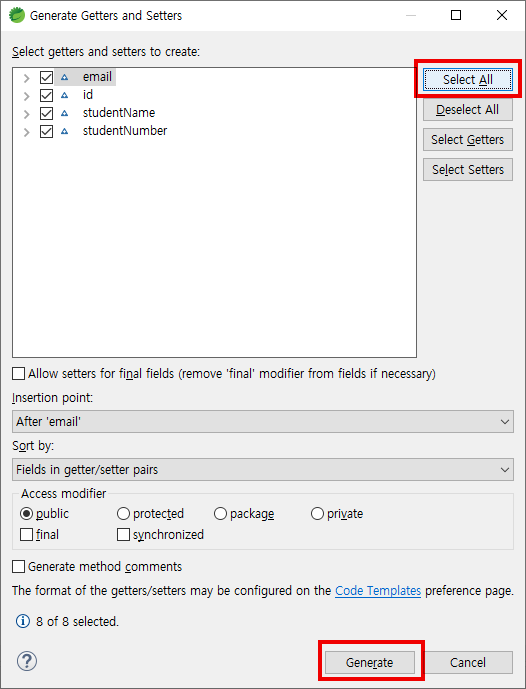
### get / set 메소드 생성

멤버 변수들 각각에 대한 get / set 메소드 생성

멤버 변수만 입력한 다음에

Source - Generate Getters and Setters 메뉴를 클릭하자.





get / set 메소드들이 자동으로 생성된다.

## ThirdController 클래스 생성

### test1 액션 메소드 구현

Student 객체를 생성한다.

Student 객체에 id, studentNumber, studentName, email 값을 대입

id 변수에는 아무 값이나 채우고, 나머지 멤버 변수에는 자신의 학번,이름, 이메일 주소를 채운다.

Model 객체에 student 객체를 넣는다.

## third/test1.jsp 생성

전달된 student 객체의 id, studnetNumber, studentName, email 값을

<table> 태그를 사용해서 출력하라.

## URL

test1 액션 메소드를 호출하기 위한 URL은 다음과 같아야 한다.

http://localhost:8088/third/test1