Spring Boot 시작하기

2021-08-31

이승진

**목차**

[1. hello 프로젝트 2](#_Toc81374774)

[1) 프로젝트 생성 2](#_Toc81374775)

[2) pom.xml 파일 수정 4](#_Toc81374776)

[3) 주의 사항 5](#_Toc81374777)

[4) application.properties 6](#_Toc81374778)

[5) HomeController.java 7](#_Toc81374779)

[6) index.jsp 9](#_Toc81374780)

[7) 실행 12](#_Toc81374781)

[2. hello 프로젝트 수정 13](#_Toc81374782)

[1) HelloController.java 수정 13](#_Toc81374783)

[2) index.jsp 수정 13](#_Toc81374784)

[3) 실행 14](#_Toc81374785)

[3. Spring Web MVC 15](#_Toc81374786)

[1) 컨트롤러 클래스 15](#_Toc81374787)

[2) 뷰(view) 17](#_Toc81374788)

[3) HelloController.java 19](#_Toc81374789)

[4) index.jsp 19](#_Toc81374790)

[5) 서버 종료하기 20](#_Toc81374791)

[4. 연습 문제 21](#_Toc81374792)

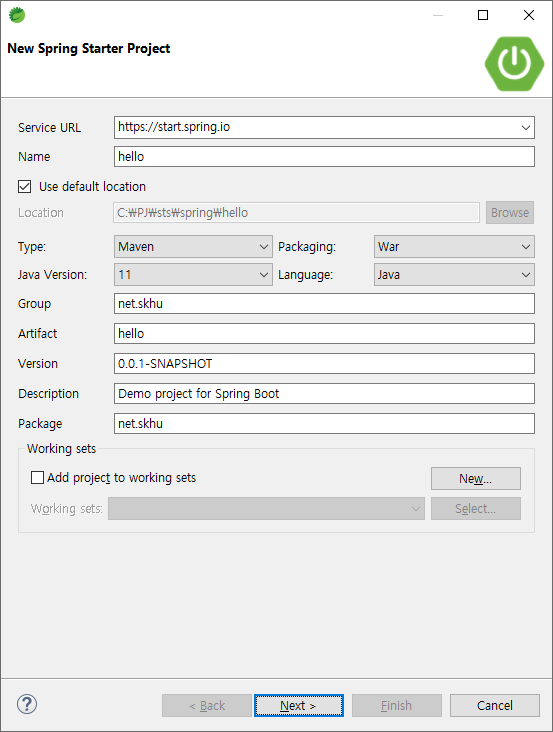
[1) 404 Not Found 에러의 원인은? 21](#_Toc81374793)

[2) model attribute 이름 21](#_Toc81374794)

# hello 프로젝트

## 프로젝트 생성

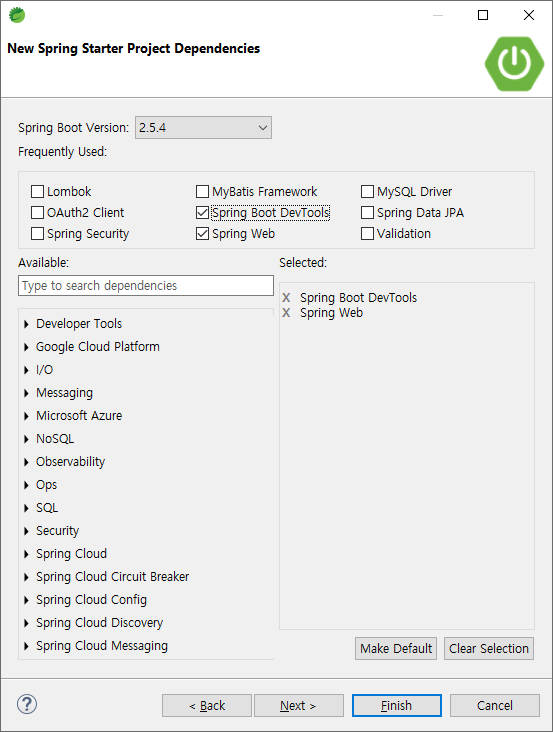
STS의 메뉴: File - New - Spring Starter Project



Name: hello

Packaing: War

Package: net.skhu



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:

Spring Web MVC

### Spring Boot DevTools 항목:

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

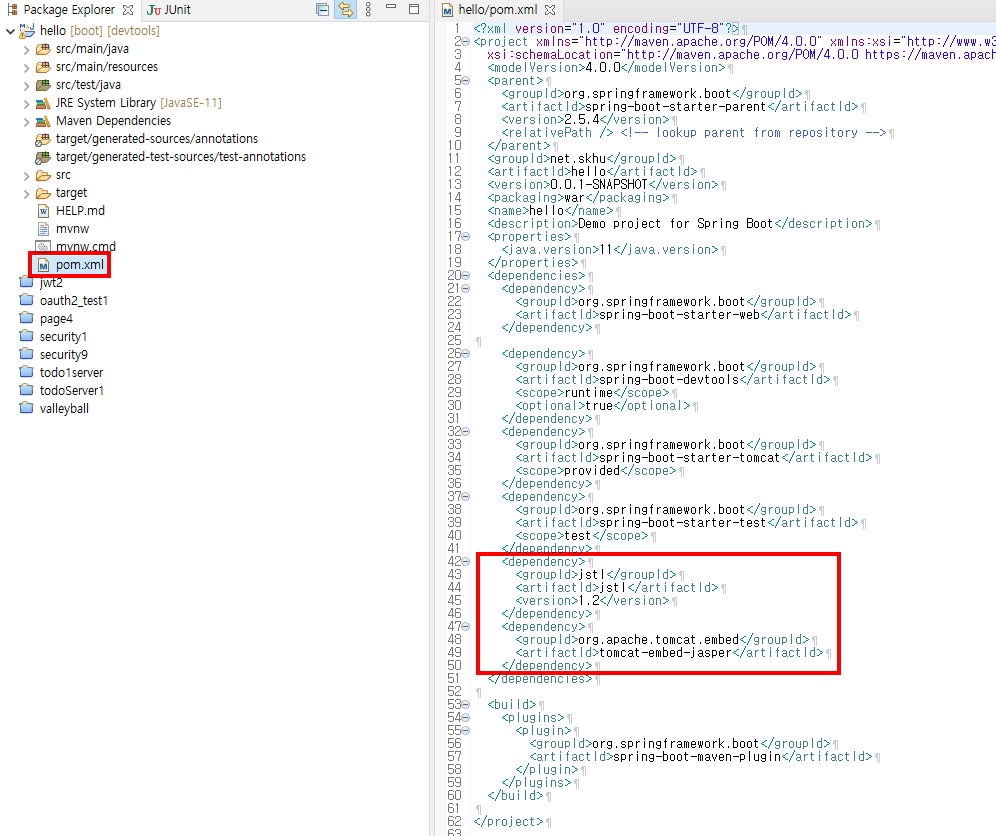
Finish 버튼을 클릭하자.

## 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 파일의 dependencies 태그에 위 내용을 추가하자.

pom.xml 파일은 maven의 설정 파일



추가된 maven dependency 항목은 다음과 같은 라이브러리를 프로젝트에 포함하기 위함이다.

### 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.

## 주의 사항

강의 노트의 소스코드를 copy & paste 할 때,

소스코드를 웹브라우저 창에서 copy 하면, 공백처럼 보이는 이상한 문자가 포함되어,

실행 오류가 발생할 수도 있다. (언제나 발생하는 것은 아니다)

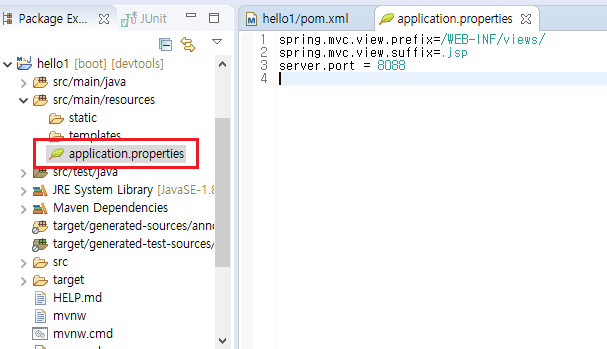
|  |
| --- |
| 이렇게 경고해도, 이 실행 오류 때문에,  구현 시험을 망치는 학생이 매 학기마다 한 두 명씩 꼭 있다.  이번 학기의 그 바보는 누구일까? |

## application.properties

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.

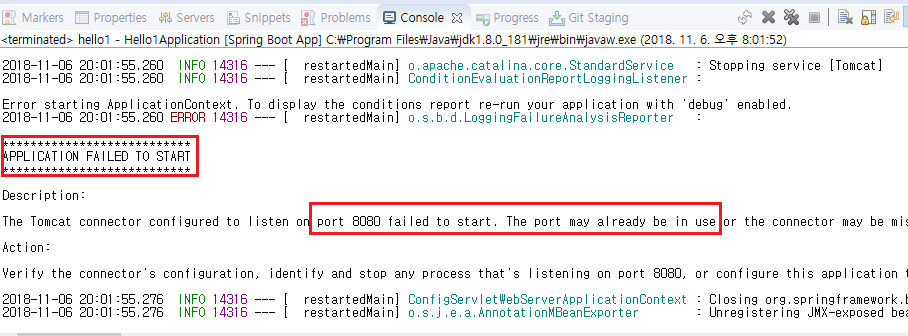
8088 포트로 변경하자.

server.port = 8088 -> 톰캣 서버가 사용할 포트 번호

### 포트 충돌 에러

포트 번호 변경하지 않고 hello 프로젝트를 실행하는 경우에,

8080 포트를 이미 다른 서버가 사용하고 있다면, 다음과 같은 포트 충돌 에러가 발생한다.



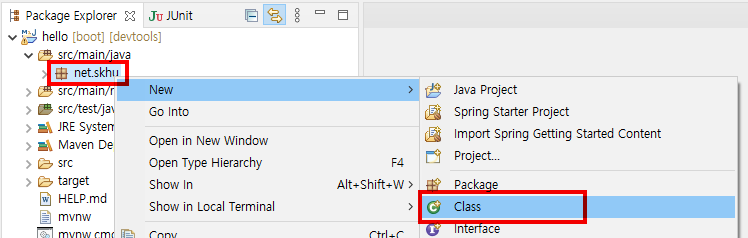
8080 포트가 이미 사용중이라는 에러 발생.

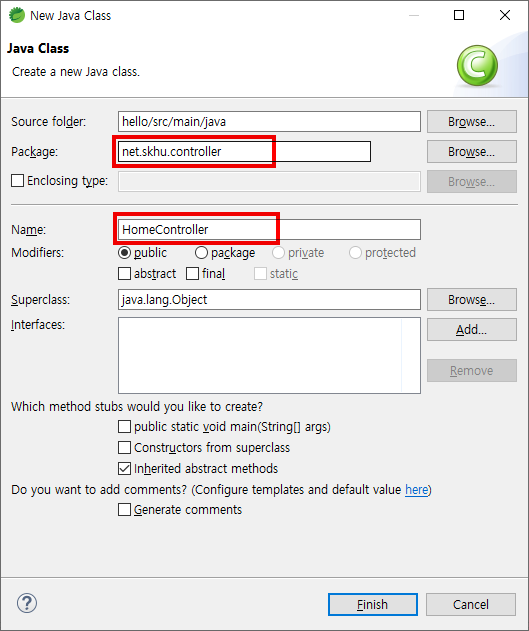
### 주의 사항

application.properties 파일의 각 줄 끝에 공백 문자가 있으면 안된다.

실행 오류가 발생한다.

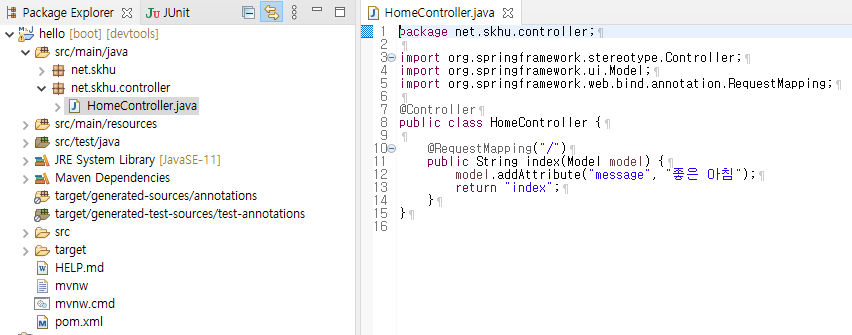
## HomeController.java



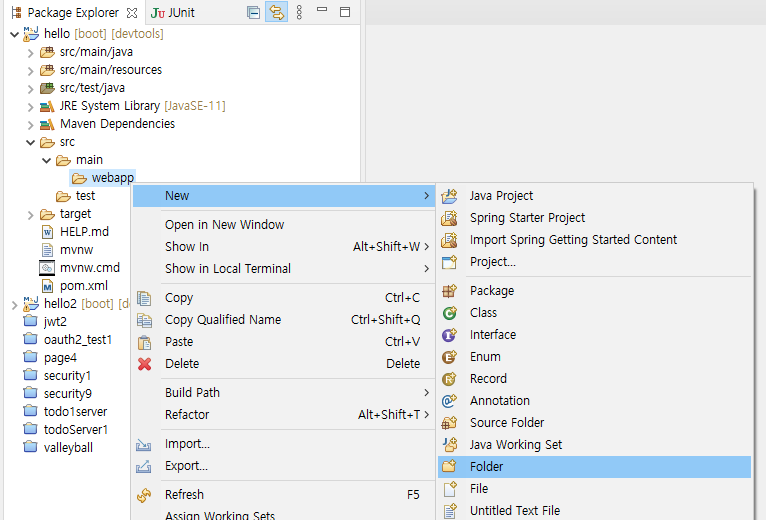


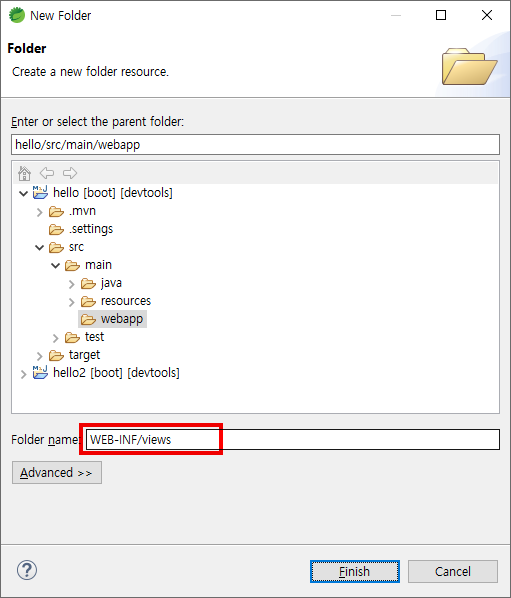
src/main/java/net/skhu/controller/HomeController.java

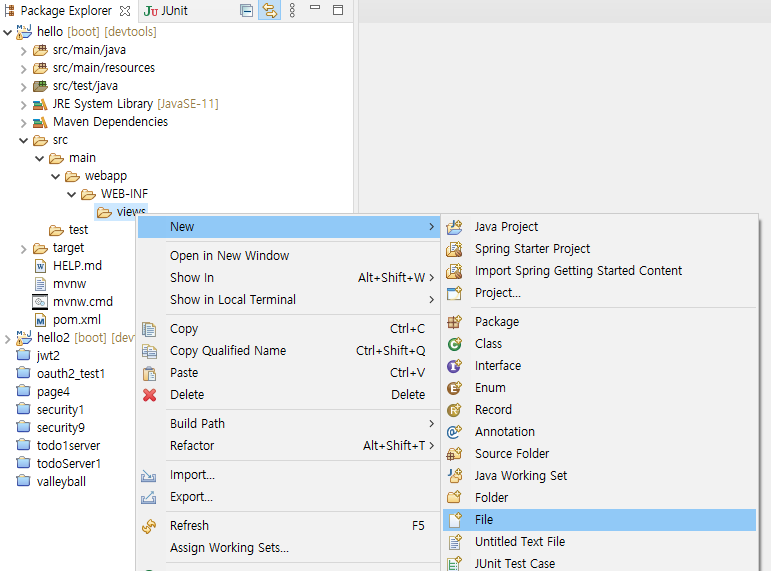
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | package net.skhu.controller;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.RequestMapping;  @Controller  public class HomeController {  @RequestMapping("/")  public String index(Model model) {  model.addAttribute("message", "좋은 아침");  return "index";  }  } |

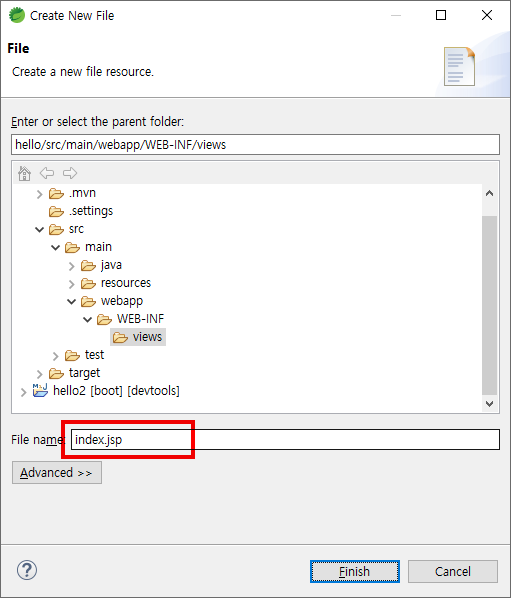


## index.jsp



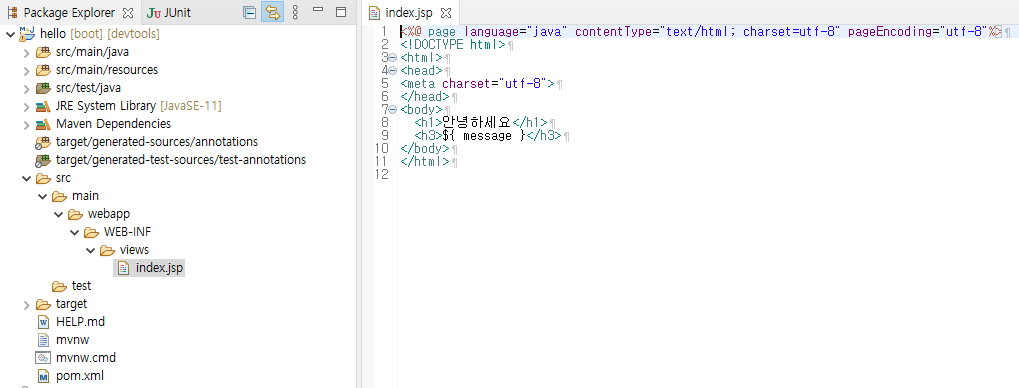






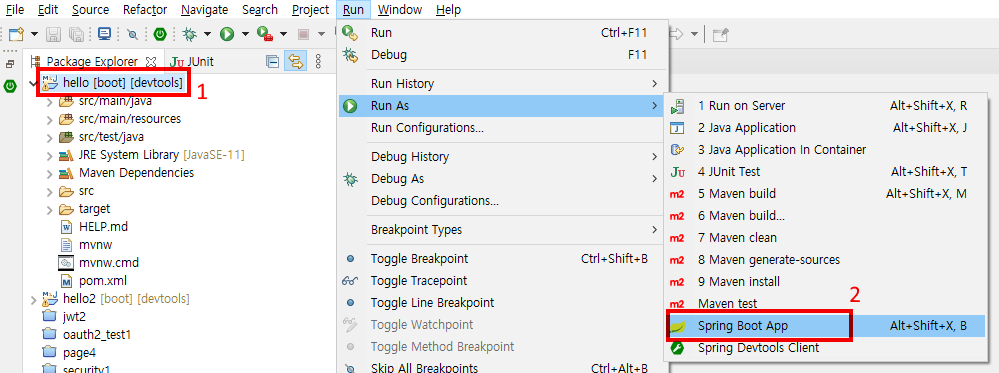
src/main/webapp/WEB-INF/views/index.jsp

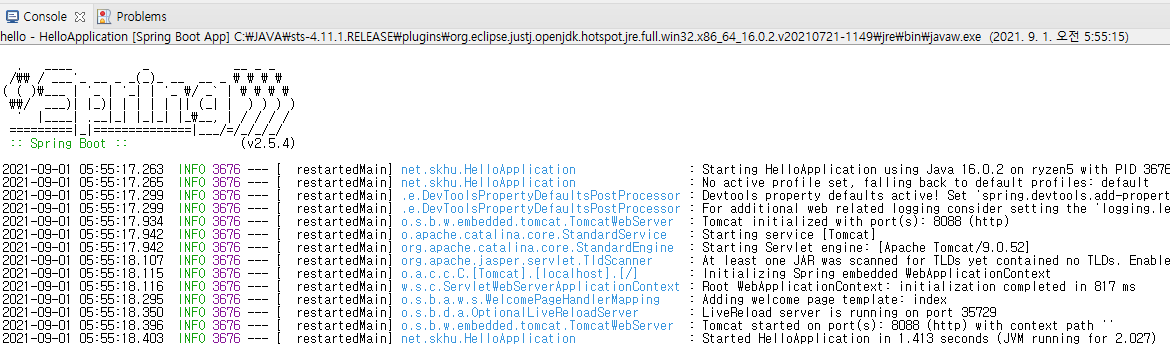
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  </head>  <body>  <h1>안녕하세요</h1>  <h3>${ message }</h3>  </body>  </html> |



## 실행

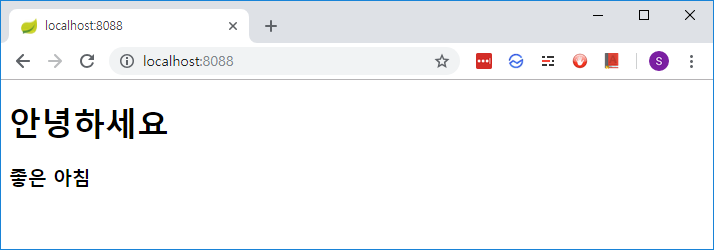
### Spring Boot App 실행





콘솔 창의 출력이 위와 같다면, Spring boot app이 잘 실행된 것이다.

웹브라우저를 열어서 http://localhost:8088 URL을 실행하자.



hello 프로젝트에는 tomcat 서버가 내장되어 있고,

이 내장된 서버가 실행중이다.

# hello 프로젝트 수정

수정된 소스코드가 얼마나 빨리 실행되는지 확인해 보자.

## HelloController.java 수정

src/main/java/net/skhu/controller/HelloController.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | package net.skhu.controller;  import java.util.Date;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.RequestMapping;  @Controller  public class HomeController {  @RequestMapping("/")  public String index(Model model) {  model.addAttribute("message", "좋은 아침");  model.addAttribute("now", new Date());  return "index";  }  } |

## index.jsp 수정

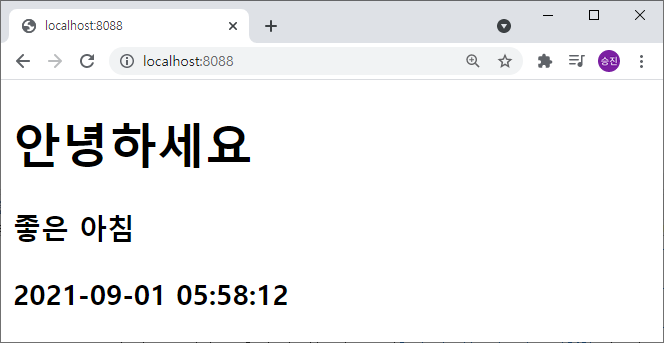
src/main/webapp/WEB-INF/views/index.jsp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <%@ 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 charset="utf-8">  </head>  <body>  <h1>안녕하세요</h1>  <h3>${ message }</h3>  <h3><fmt:formatDate pattern = "yyyy-MM-dd HH:mm:ss" value = "${ now }" /></h3>  </body>  </html> |

## 실행

소스코드를 수정한 다음에, 웹브라우저 창에서 새로고침하면, 수정된 부분이 반영되어 다시 실행된다.

웹브라우저를 새로고침하자.



수정된 내용이 꽤 빨리 반영된다.

# Spring Web MVC

## 컨트롤러 클래스

### HomeController.java

|  |
| --- |
| package net.skhu.controller;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.RequestMapping;  @Controller  public class HomeController {  @RequestMapping("/")  public String index(Model model) {  model.addAttribute("message", "좋은 아침");  return "index";  }  } |

컨트롤러 클래스는 웹브라우저의 URL 요청을 받아서, 웹서버에서 실행되는 자바 클래스이다.

웹브라우저가 웹서버에 어떤 URL을 요청을 하면, 그 URL에 해당하는 컨트롤러의 메소드가 자동으로 호출되어 실행된다.

### @Controller

컨트롤러 클래스에 이 어노테이션(annotation)을 붙여주어야 한다.

이것을 붙여주지 않으면 컨트롤러 클래스가 실행되지 않는다. 404 Not Found 에러가 발생할 것이다.

### 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

액션 메소드에 붙은 @RequestMapping("**/**") 어노테이션의 "**/**" 부분은

그 액션 메소드를 호출하기 위한 URL 이다.

**http://localhost:8080/** URL을 웹브라우저가 웹서버에 요청하면,

웹서버에서 HomeController 컨트롤러 클래스의 index 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.

만약 요청된 URL과 일치하는 액션 메소드가 없다면, 404 Not Found 에러가 발생할 것이다.

### 컨트롤러 액션 메소드의 리턴 값

컨트롤러의 액션 메소드는 문자열을 리턴한다.

컨트롤러의 액션 메소드가 리턴하는 문자열은, 뷰(View) 파일의 이름이다.

컨트롤러의 액션 메소드가 리턴된 후, 뷰 파일이 실행된다.

액션 메소드가 리턴한 이름의 뷰 파일이 실행된다.

### Model 객체

|  |
| --- |
| public String index(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.

### model attribute

|  |
| --- |
| model.addAttribute("message", "좋은 아침"); |

뷰 파일에 전달할 데이터를 Model 객체에 넣는 코드이다.

전달되는 데이터의 이름은 "message" 이고, 값은 "좋은 아침" 문자열이다.

이렇게 모델에 담겨서 뷰에 전달되는 데이터를 model attribute 이라고 한다.

뷰에서 model attribute를 출력하는 코드는 아래와 같다.

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

model attribute 이름이 정확히 일치해야 한다.

## 뷰(view)

index.jsp

|  |
| --- |
| <%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  </head>  <body>  <h1>안녕하세요</h1>  <h3>${ message }</h3>  </body>  </html> |

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

웹서버에서 웹브라우저로 전송되는 것은 대부분 html 태그이다.

우리가 웹서핑하면서 보는 웹페이지들의 내용이 html 태그이다.

### 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.

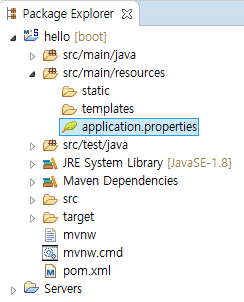
### 뷰 파일 위치 설정

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



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

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

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 "index"; |

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

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

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

|  |
| --- |
| <%@ 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)

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

index.jsp 뷰 파일에 전달된 데이터 중에서, 이름이 "message"인 데이터를 여기에 출력하라는 명령이다.

## HelloController.java

src/main/java/net/skhu/HelloController.java

|  |
| --- |
| package net.skhu.controller;  import java.util.Date;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.RequestMapping;  @Controller  public class HomeController {  @RequestMapping("/")  public String index(Model model) {  model.addAttribute("message", "좋은 아침");  model.addAttribute("now", new Date());  return "index";  }  } |

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

뷰 파일에 전달할 데이터를 Model 객체에 넣는 코드이다.

전달되는 데이터의 이름은 "now" 이고, 값은 현재 시각 Date 객체이다.

## index.jsp

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

|  |
| --- |
| <%@ 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 charset="utf-8">  </head>  <body>  <h1>안녕하세요</h1>  <h3>${ message }</h3>  <h3><fmt:formatDate pattern = "yyyy-MM-dd HH:mm:ss" value = "${ now }" /></h3>  </body>  </html> |

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

<fmt:formatDate /> 태그는 날짜와 시각을 포멧에 맞춰서 출력하기 위한 JSTL 확장 태그이다.

JSTL 확장 태그를 사용하기 위해 pom.xml 파일에 JSTL dependency를 추가했었다.

value = "${ now }"

뷰 파일에 전달된 데이터 중에서, 이름이 "now"인 데이터가 출력된다.

pattern = "yyyy-MM-dd HH:mm:ss"

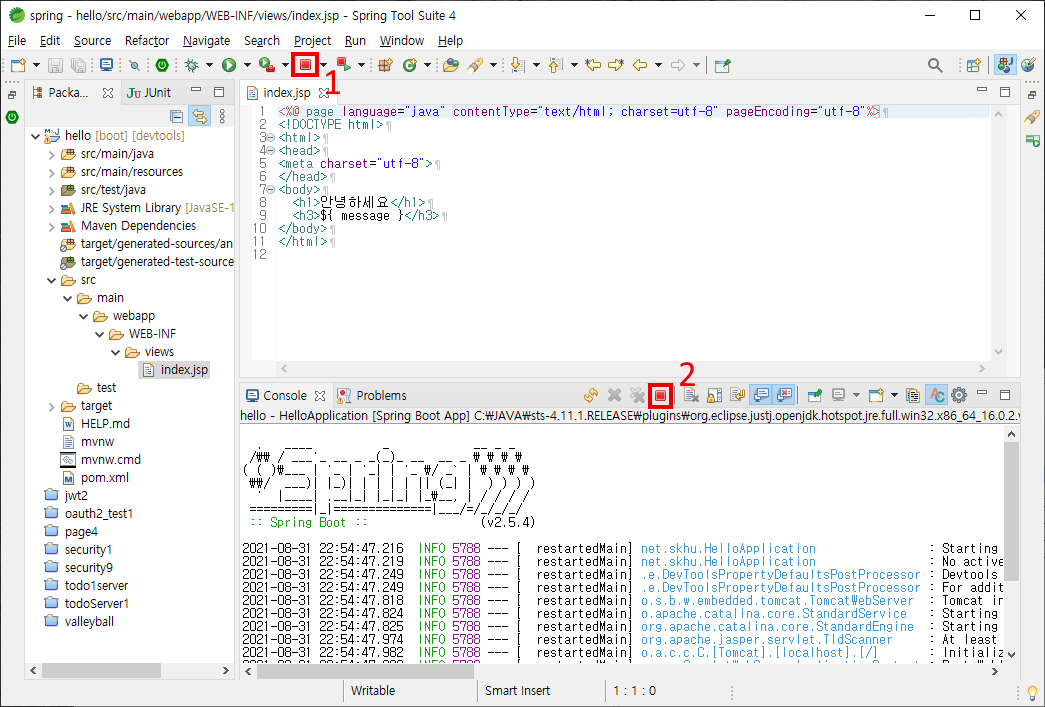
날짜와 시각이 이 포멧에 맞춰서 출력된다. 이 포멧 문자열은 Java의 SimpleDateFormat 클래스의 그것이다.

|  |
| --- |
| <%@ taglib prefix = "fmt" uri = "http://java.sun.com/jsp/jstl/fmt" %> |

<fmt:formatDate /> 확장 태그를 사용하려면, 먼저 이 선언이 있어야 한다.

## 서버 종료하기

서버를 종료하자.



1번이나 2번 버튼이 활성화되어 있다면, 서버가 실행 상태라는 것을 의미한다.

버튼을 클릭하여 서버를 종료하자.

# 연습 문제

## 404 Not Found 에러의 원인은?

프로젝트를 실행하고, 웹브라우저 창에서 URL을 입력했더니, 404 Not Found 에러가 발생했다.

이 에러의 원인을 찾아서 고치기 위해 확인해야 할 부분을 모두 나열하라.

## model attribute 이름

index.jsp 에서 model attribute 이름을 일부러 틀리게 수정하고 실행해 보자.

|  |
| --- |
| <h3>${ message }</h3> |

위 부분을 다음과 같이 수정하자.

|  |
| --- |
| <h3>${ msg }</h3> |

model attribuet 이름이 틀렸을 때, 어떤 증상이 발생하는가?