



Seoul  
Software  
ACademy

# 웹 개발자 부트캠프 과정

SeSAC x CODINGOn

With. 팀 뽀빠

# Spring Boot 프로젝트 생성하기

# Spring Boot 프로젝트 시작하기

<https://start.spring.io/>

- Spring Boot 기반으로 Spring 관련 프로젝트를 만들어주는 사이트
- Spring에서 운영하고 있으며, 해당 사이트에서 원하는 라이브러리를 선택 후 프로젝트 생성 가능

# Spring Boot 프로젝트 시작하기

spring initializr

Project

☒ Gradle - Groovy
 ☐ Gradle - Kotlin
 ☐ Maven

Language

☒ Java
 ☐ Kotlin
 ☐ Groovy

Spring Boot

☐ 3.2.1 (SNAPSHOT)
 ☒ 3.2.0
 ☐ 3.1.7 (SNAPSHOT)
 ☐ 3.1.6

Project Metadata

Group

com.example

Artifact

demo

Name

demo

Description

Demo project for Spring Boot

Package name

com.example.demo

Packaging

☒ Jar
 ☐ War

Java

☐ 21
 ☒ 17

Dependencies

ADD DEPENDENCIES... CTRL + B

No dependency selected

GENERATE CTRL + G

EXPLORE CTRL + SPACE

SHARE...



생성할 프로젝트의 빌드 자동화 툴

#### Project

☒ Gradle - Groovy

☐ Gradle - Kotlin

☐ Maven

스프링 부트 버전

#### Spring Boot

☐ 3.2.1 (SNAPSHOT) ☒ 3.2.0 ☐ 3.1.7 (SNAPSHOT)

☐ 3.1.6

프로젝트 정보

#### Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 21 ☒ 17

생성할 프로젝트의 언어

#### Language

☒ Java

☐ Kotlin

☐ Groovy

프로젝트 의존성

여기에 추가한 디펜던시가 **build.gradle** 파일에 추가됨!

#### Dependencies

**ADD DEPENDENCIES...** CTRL + B

#### Spring Web

**WEB**

스프링 기반으로 웹 개발

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

#### Thymeleaf

**TEMPLATE ENGINES**

템플릿 엔진

A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

#### Lombok

**DEVELOPER TOOLS**

Java annotation library which helps to reduce boilerplate code.

코드 다이어트 라이브러리

ex. getter, setter 등 코드를 간단히 사용하게 함



**GENERATE** CTRL + G

**EXPLORE** CTRL + SPACE


**SHARE...**

# 프로젝트 압축 파일 프로젝트 경로로 옮기기

내 PC > 다운로드

이름

오늘 (5)


sesac-spring-boot.zip

내 PC > 새 볼륨 (D:) > 공부용\_프로젝트\_폴더

이름

.idea

sesac-spring-boot

sesac-spring-boot >

이름

.gradle

gradle

src

.gitignore

build.gradle

gradlew

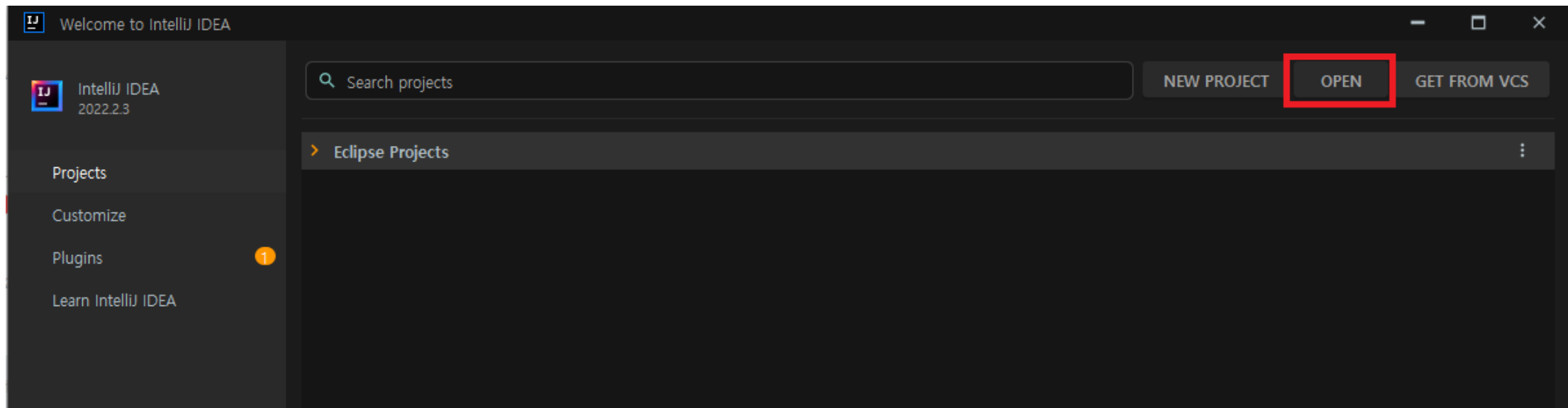
gradlew.bat

HELP.md

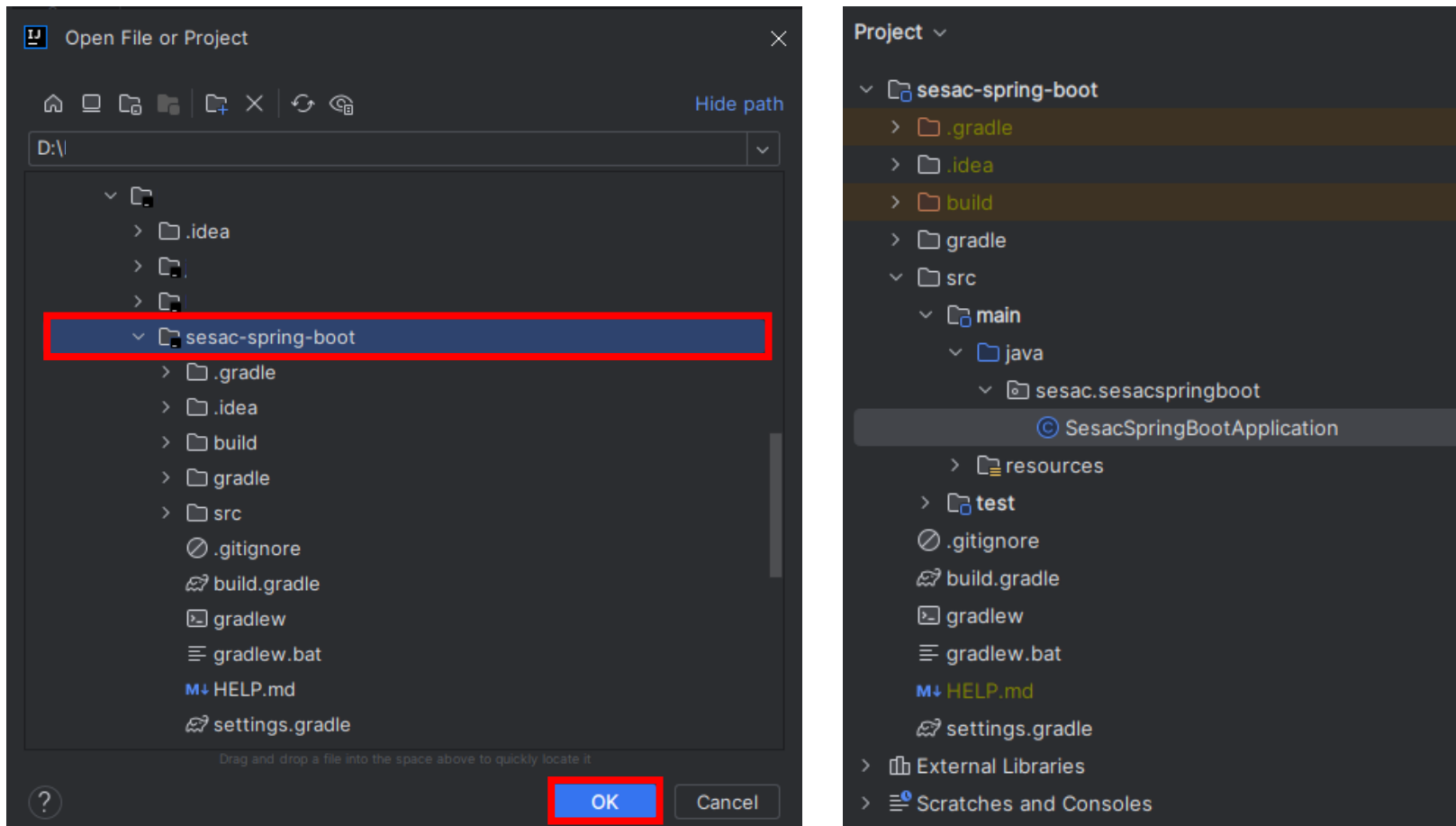
settings.gradle

압축 풀어서 옮기기~

# IntelliJ 에서 프로젝트 열기



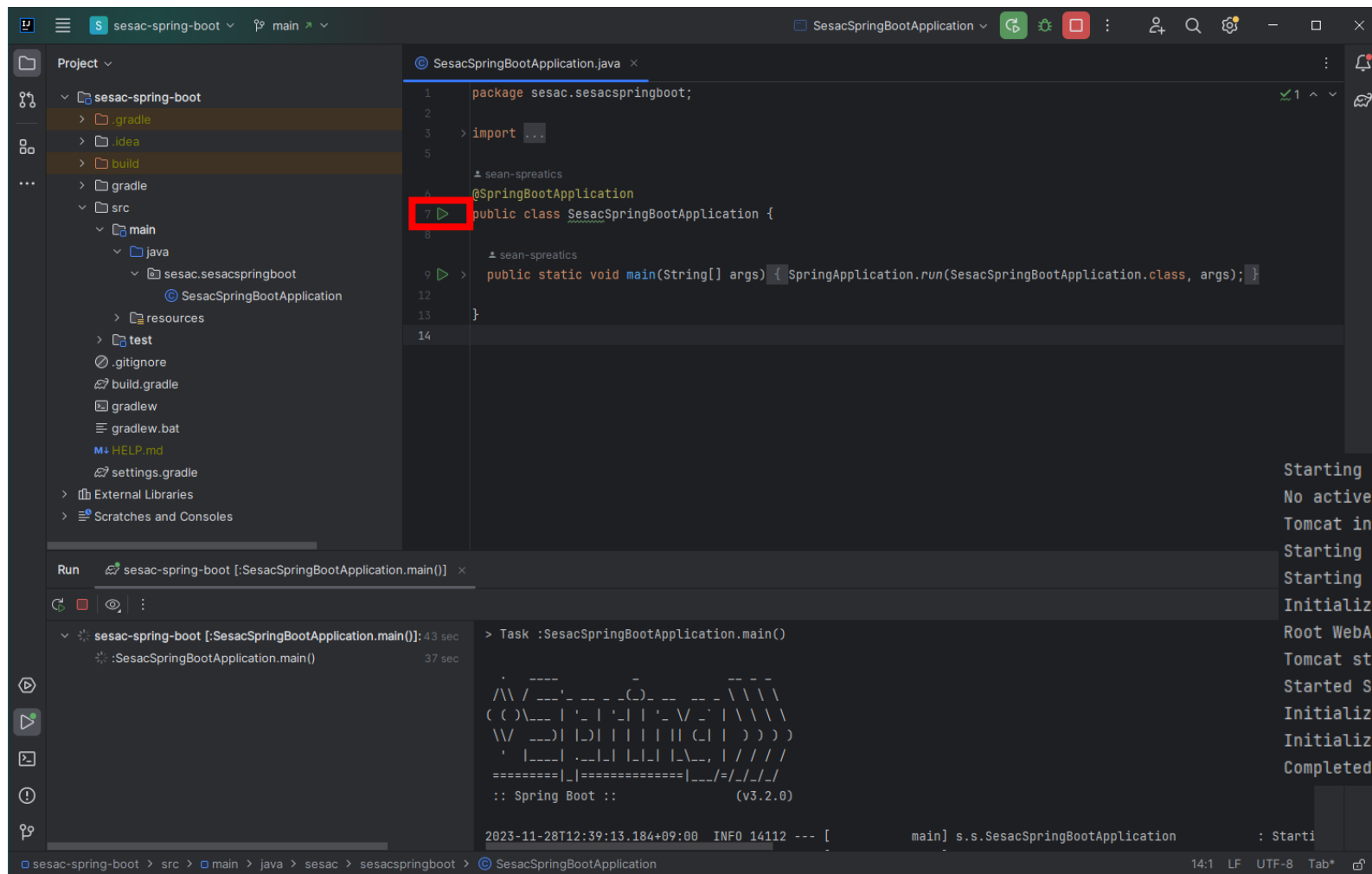
# IntelliJ 에서 프로젝트 열기



주의. src/ 폴더가 프로젝트이름 바로 아래에 위치하도록 OPEN 하기!



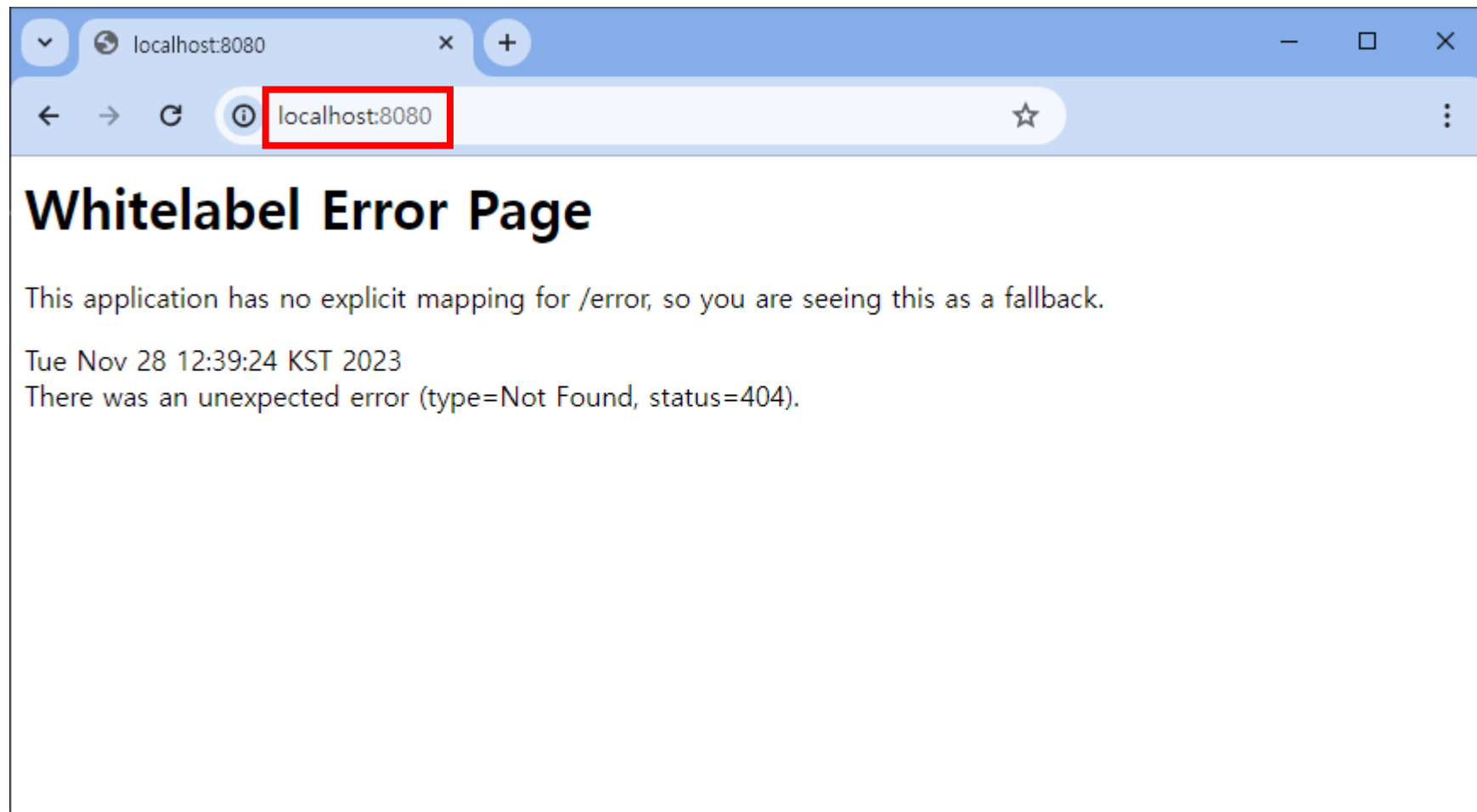
# 프로젝트 실행하기



스프링 부트는 8080 포트 번호를 사용해  
내장 웹 서버를 실행함!

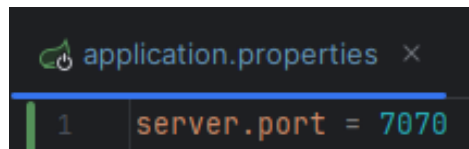
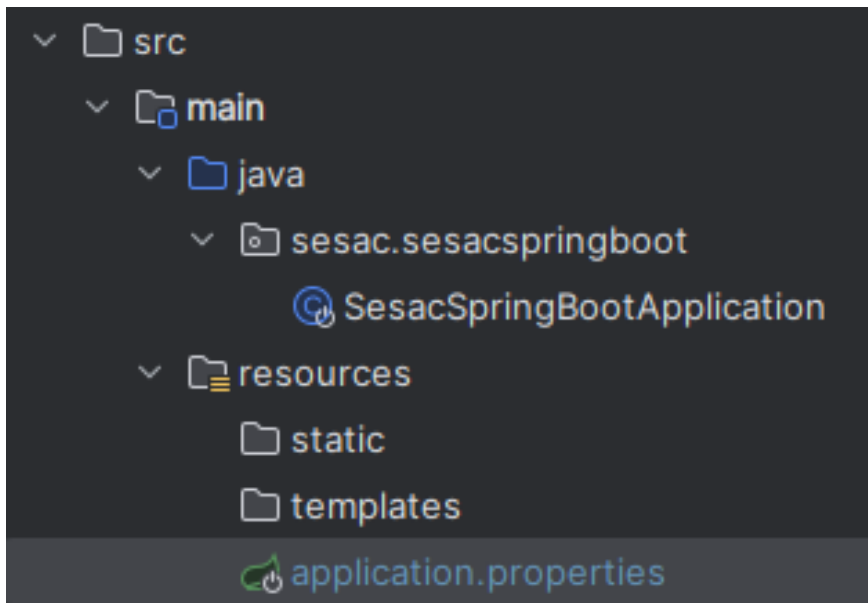
```
Starting SesacSpringBootApplication using Java 17.0.8 with PID 14112
No active profile set, falling back to 1 default profile: "default"
Tomcat initialized with port 8080 (http)
Starting service [Tomcat]
Starting Servlet engine: [Apache Tomcat/10.1.16]
Initializing Spring embedded WebApplicationContext
Root WebApplicationContext: initialization completed in 857 ms
Tomcat started on port 8080 (http) with context path ''
Started SesacSpringBootApplication in 1.502 seconds (process running for 1.769)
Initializing Spring DispatcherServlet 'dispatcherServlet'
Initializing Servlet 'dispatcherServlet'
Completed initialization in 0 ms
```

# 브라우저 접속하기



# 참고. 서버 포트번호 변경하기

- 스프링 부트는 기본적으로 8080 포트번호를 사용해 내장 웹 서버를 실행
- 포트번호를 변경하고 싶다면, `application.properties` 파일에서 별도 설정 필요



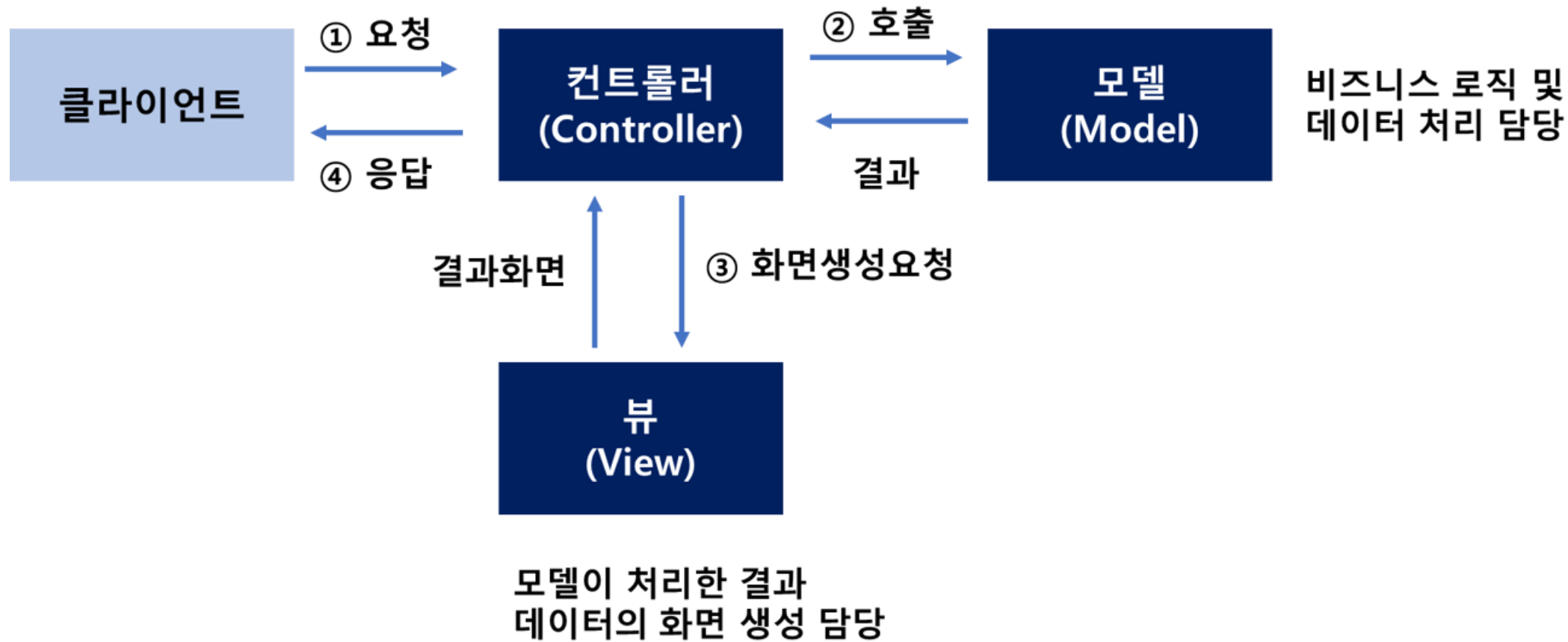
```
Tomcat initialized with port 7070 (http)
Starting service [Tomcat]
Starting Servlet engine: [Apache Tomcat/10.1.16]
Initializing Spring embedded WebApplicationContext
Root WebApplicationContext: initialization completed in 821 ms
Tomcat started on port 7070 (http) with context path ''
Started SesacSpringBootApplication in 1.488 seconds (process running for 1.794)
```

# 프로젝트 구조 살펴보기

- main/ : 실제 코드를 작성하는 곳
- test/: 프로젝트의 소스 코드를 테스트하는 코드 or 리소스 파일
- build.gradle: 빌드 설정 파일
- setting.gradle: 빌드할 프로젝트 정보 설정 파일

# Spring MVC

# Spring MVC



# Spring MVC

- Spring 에서 제공하는 웹 모듈로 웹 애플리케이션을 빌드하기 위한 프레임워크
- Model, View, Controller 의 3가지 구성 요소
  - Model: 데이터와 비즈니스 로직 처리
  - View: 사용자에게 보여지는 부분
  - Controller: 사용자의 요청을 받아 처리하고 적절한 Model 을 호출한 후, 그 결과를 View 에게 전달
- MVC 패턴을 지원하고 DispatcherServlet 이라는 특수 서블릿을 통해 요청 처리
  - 이 서블릿은 모든 종류의 요청을 받아 적절한 Controller 에게 전달하고 그 결과를 다시 사용자에게 반환

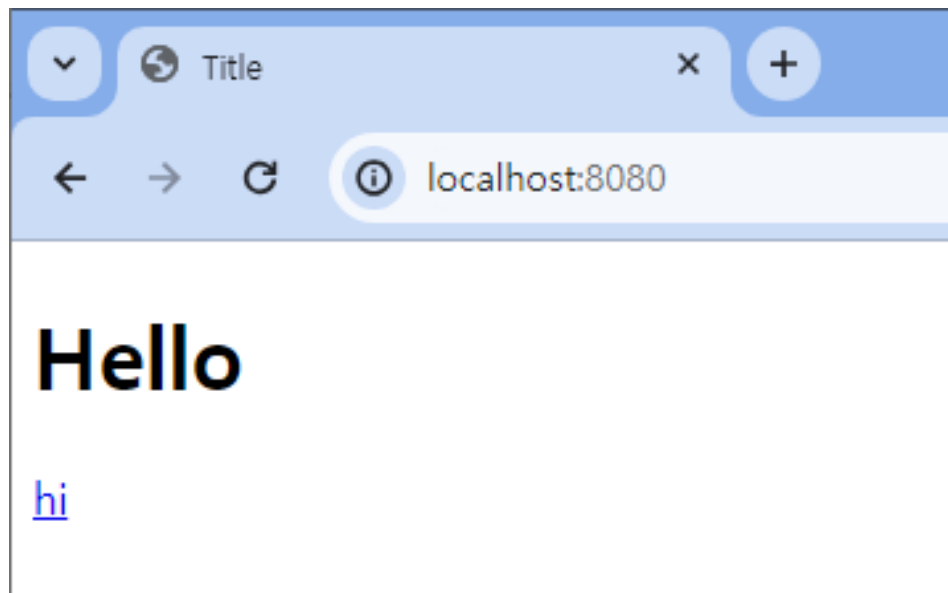
# 정적 파일



# 실행 화면 보여주기

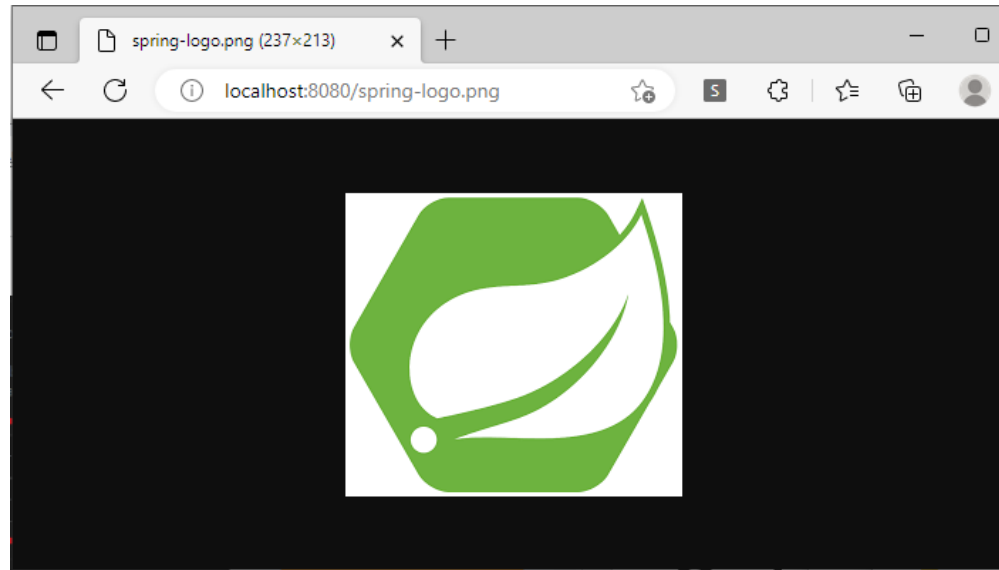
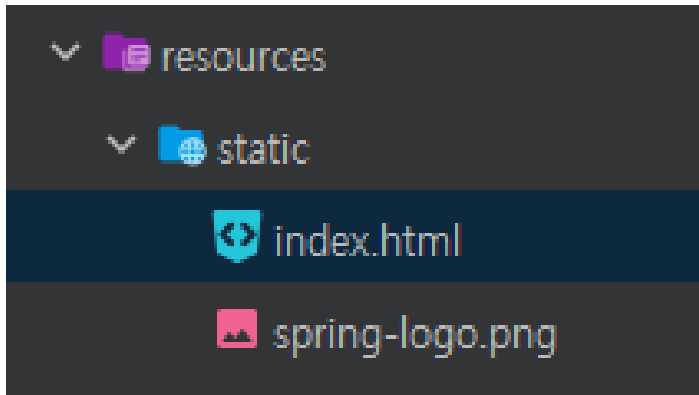
- ./src/main/resources/static에 index.html 생성

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <h1>Hello</h1>
  <a href="/hi">hi</a>
</body>
</html>
```



# 정적 파일

- ./src/main/resources/static 는 정적 파일 위치
- 위치하고 있는 파일을 보여준다



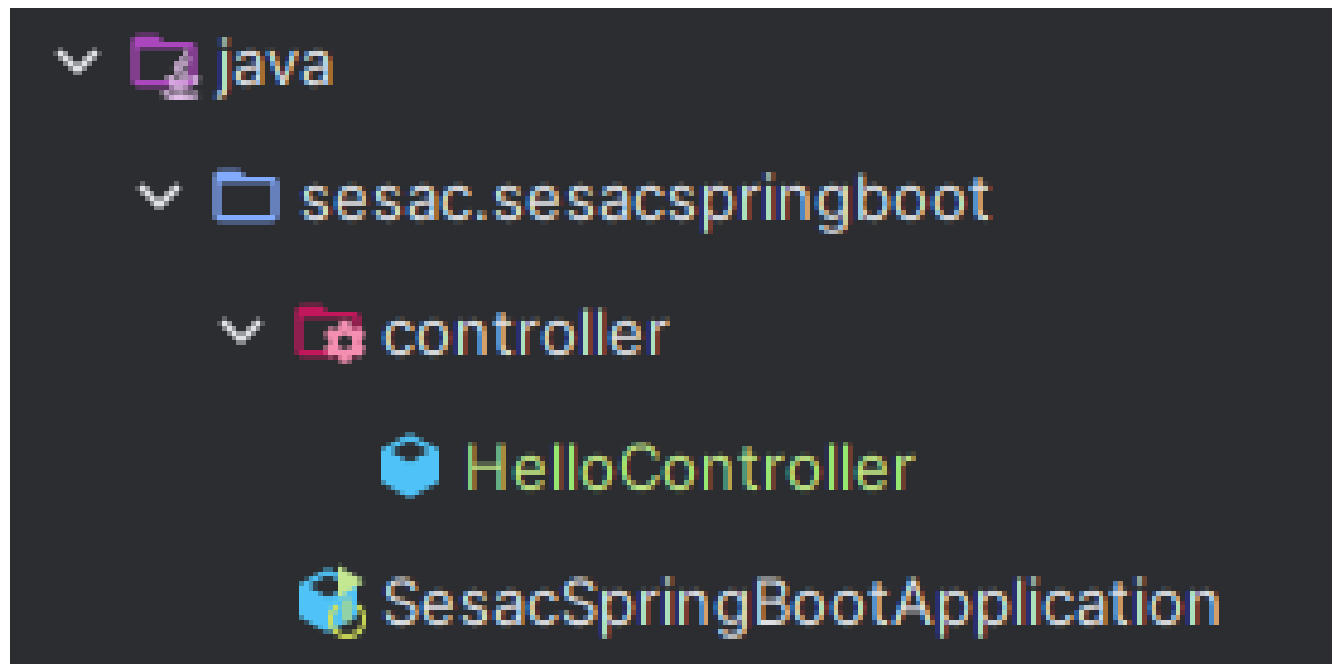
# Thymeleaf 템플릿

# Thymeleaf 란?

- 템플릿 엔진
  - ex. JSP, Thymeleaf, FreeMarker, ...
- HTML 태그에 속성을 추가해 페이지에 동적으로 값을 추가하거나 처리할 수 있게 도와주는 것
- Spring Boot 사용시 권장되는 템플릿 엔진



# Controller 만들기



# Controller 만들기

```
package sesac.sesacspringboot.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class HelloController {
    @GetMapping("/hi")
    public String getHi(Model model) {
        model.addAttribute("msg", "Hi~");
        return "hi";
    }
}
```

## @Controller

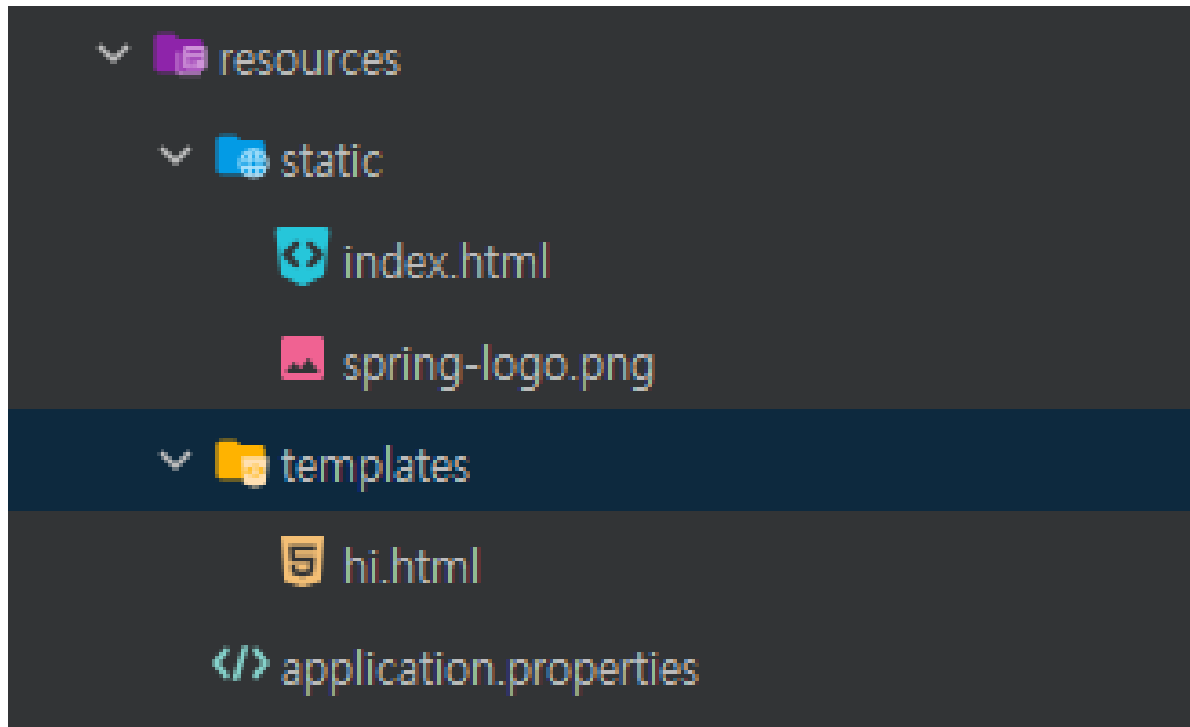
: 해당 클래스가 Controller 클래스라는 것을 Spring Container에게 알려준다.

## @GetMapping

: URL을 매핑시켜주는 것으로 get method로 해당 경로로 들어올 시 getHi라는 함수를 실행시킨다.

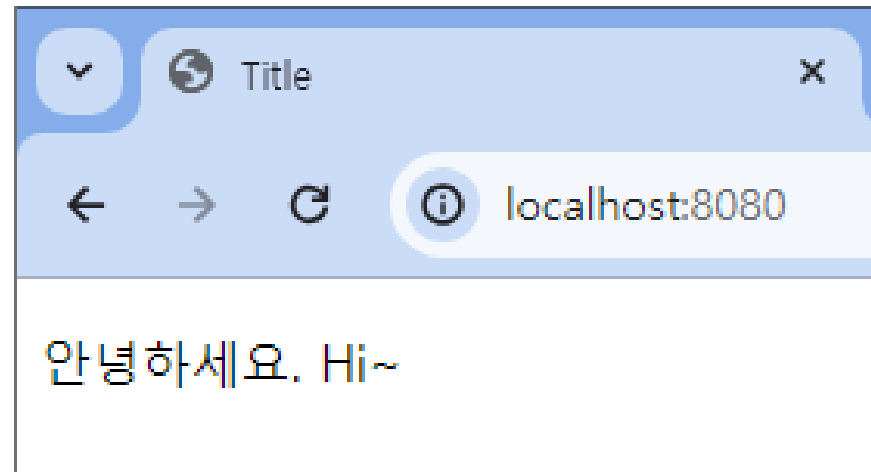
참고. @XXX 이런 친구들은 “Annotation” 단원에서 자세히 다룰 예정!

# Template view 만들기



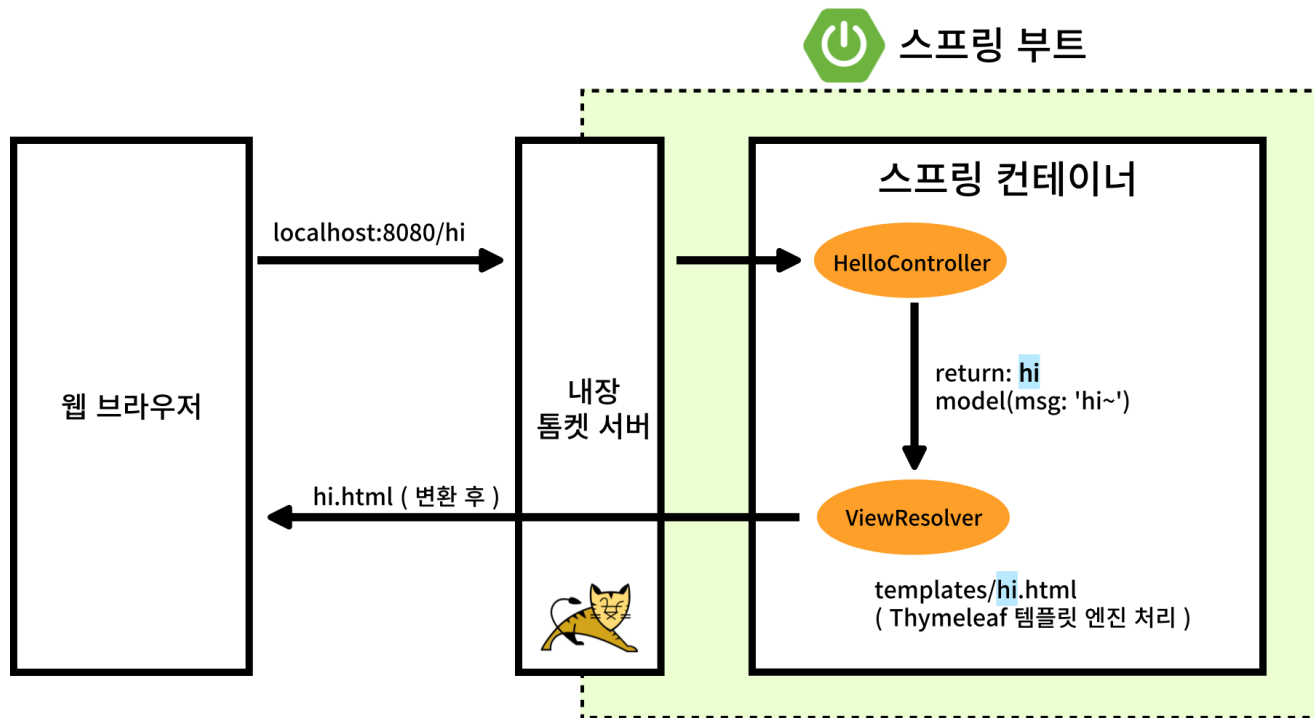
# Template view 만들기

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <p th:text="'안녕하세요.' + ${msg}">안녕 반가워</p>
</body>
</html>
```





# Spring Boot 동작 환경



스프링 부트는 WAS로 Tomcat을 내장

1. localhost:8080/hi 요청 시, 톰캣이 스프링한테 넘겨줌
2. 컨트롤러(HelloController.java)에서 매칭되는 URL 을 찾음
3. 해당 메소드(getHi) 에서 반환하는 문자열을 ViewResolver 가 화면을 찾아서 처리 (templates/hi.html)

# Thymeleaf

## 표현식과 문법

# Thymeleaf 표현식

- 서버에서 전달받은 데이터를 사용자들이 볼 수 있는 뷰로 만들기 위해 사용되는 표현식

표현식	설명
<code>\${...}</code>	변수의 값 표현식
<code>#{...}</code>	속성 파일 값 표현식
<code>@{...}</code>	URL 표현식
<code>*{...}</code>	선택한 변수의 표현식 th:object 에서 선택한 객체에 접근

# Thymeleaf 문법

- HTML 태그 안에 소 문법을 추가
- div 태그 뿐만 아니라 HTML 에서 지원하는 태그들 모두 사용 가능

```
<div th:[속성]="서버에서 받는 값 및 조건식" />
```

# th:text

- 태그 안의 텍스트를 서버에서 전달받은 값에 따라 표현하고자 할 때 사용하는 문법

```
<span th:text="${hello}">message</span>
```

# th:utext

- th:text와 유사
- 변수에서 받은 값에 html 태그가 있다면 태그 값을 반영해서 표시해줌
- “<strong>안녕</strong>” 값이 서버에서 넘어왔다면 <strong>이라는 글자가 나오는 것이 아닌 “안녕”이라는 글자가 <strong> 태그에 의해 굵게 표시됨

# th:value

- HTML Element 의 value 속성 값을 지정할 때 사용

```
<button th:value="${hello}" />|
```

# th:with

- 변수 값을 지정해서 사용하고자 할 때 사용

```
<div th:with="temp=${hello}" th:text="${temp}" />
```

- 위와 같이 사용한다면 서버에서 전달받은 'hello' 값이 저장된 temp 라는 변수가만 들어지고, Thymeleaf 문법 내에서 사용 가능



# th:switch

- switch-case문을 이용할 때 사용되는 문법
- th:case 에서 case 문을 다루고, \* 로 case문에서 다루지 않은 모든 경우가 처리됨 ( \* 는 일반 switch-case 문법에서의 default )

```
<div th:switch="${hello}">
  <p th:case="'admin'">hello is admin</p>
  <p th:case="'manager'">hello is manager</p>
  <p th:case="*">hello is other</p>
</div>
```

# th:if

- 조건문이 필요할 때 사용되는 문법
- Else 문이 필요한 경우에는 th:unless 를 사용

```
<p th:if="${hello}=='web'" th:text="${hello}"></p>  
<p th:unless="${hello}=='web'" th:text="unless입니다."></p>
```

- 주의. th:unless 에 if에 적은 조건을 적어줘야 위의 if가 아닌 경우를 인식 가능!!

# th:each

- 반복문이 필요한 경우에 사용되는 문법
- 리스트와 같은 **collection 자료형**을 서버에서 넘겨주면 그에 맞춰 반복적인 작업이 이루어질 때 사용 가능

```
String[] names = {"kim", "lee", "hong", "park", "shin"};
model.addAttribute(attributeName: "names", names);
return "05_Thymeleaf";
```

```
<ul>
  <li th:each="name:${names}">
    <span th:text="${name}">이름</span>
  </li>
</ul>
```

- kim
- lee
- hong
- park
- shin

# 실습. Thymeleaf (1)

1. Controller에서 age를 보냈을 때 20세보다 적으면 아래와 같이 출력

10세는 미성년자입니다.

2. Age가 20보다 크다면 아래와 같이 출력

20세는 성인입니다.

# 실습. Thymeleaf (2)

1. Controller에서 name과 age 정보를 갖고 있는 Person 클래스 만들기
2. ArrayList에 각기 다른 정보를 갖고 있는 Person 최소 4개 넣기
3. <http://localhost:8080/people> 에 접속했을 때 최소 4명의 사람에 대한 이름과 나이를 table 로 보여주기

이름	나이
kim	10
lee	20
hong	30
park	40
shin	50