


SeSAC 4기, 

# Spring Boot 수업

WITH 팀 리처드





# Spring Boot 시작하기


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

- Spring Boot 기반으로 Spring 관련 프로젝트를 만들어주는 사이트
- Spring에서 운영하고 있으며, 해당 사이트에서 프로젝트 시작을 많이 한다.

# Spring Boot 시작하기









**Project**  
☒ Gradle Project  
☐ Maven Project

**Language**  
☒ Java ☐ Kotlin  
☐ Groovy

**Spring Boot**  
☐ 3.0.0 (SNAPSHOT) ☐ 3.0.0 (RC1) ☐ 2.7.6 (SNAPSHOT)  
☒ 2.7.5 ☐ 2.6.14 (SNAPSHOT) ☐ 2.6.13

**Project Metadata**  
Group   
Artifact   
Name   
Description   
Package name   
Packaging ☒ Jar ☐ War  
Java ☐ 19 ☒ 17 ☐ 11 ☐ 8

**Dependencies**   
*No dependency selected*



### Project

☒ Gradle Project

☐ Maven Project

### Language

☒ Java ☐ Kotlin

☐ Groovy

### Spring Boot

☐ 3.0.0 (SNAPSHOT) ☐ 3.0.0 (RC1) ☐ 2.7.6 (SNAPSHOT)

☒ 2.7.5 ☐ 2.6.14 (SNAPSHOT) ☐ 2.6.13

### Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 19 ☒ 17 ☐ 11 ☐ 8

### Dependencies

[ADD DEPENDENCIES...](#) CTRL + B

#### Spring Web WEB

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

#### Thymeleaf TEMPLATE ENGINES

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

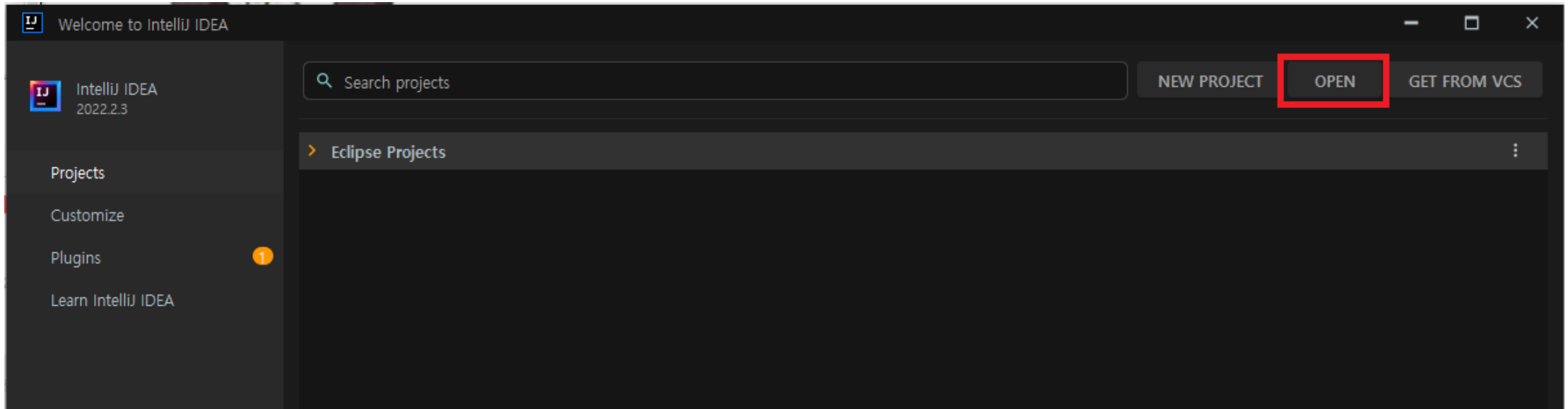
Java annotation library which helps to reduce boilerplate code.

[GENERATE](#) CTRL + G

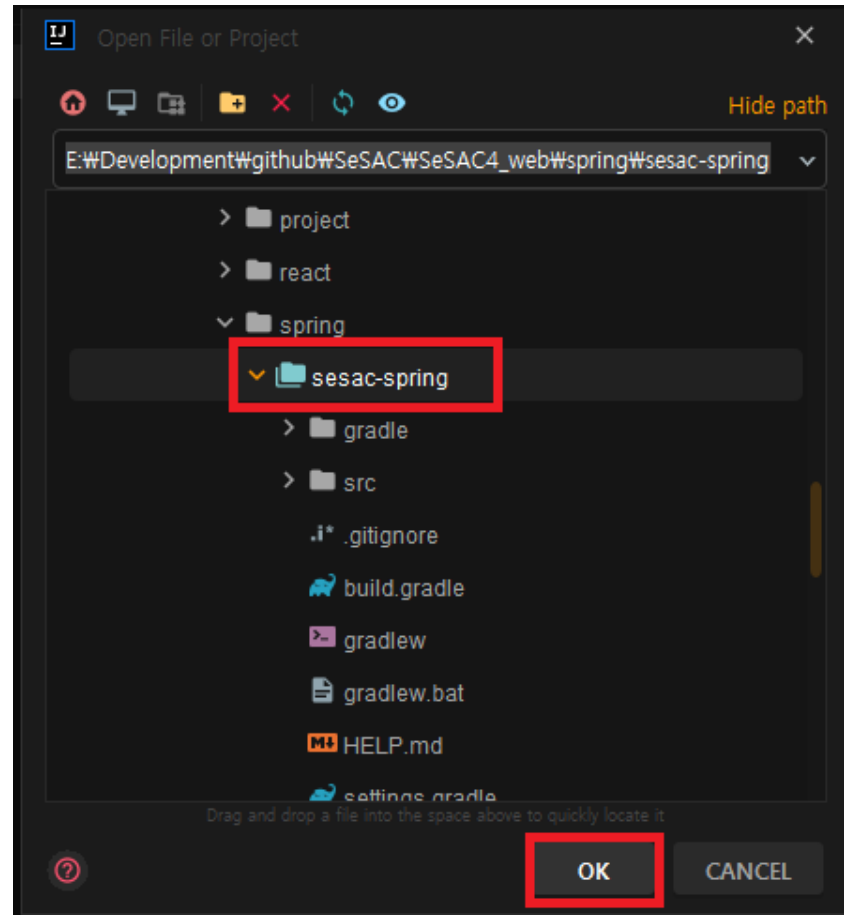
[EXPLORE](#) CTRL + SPACE

[SHARE...](#)

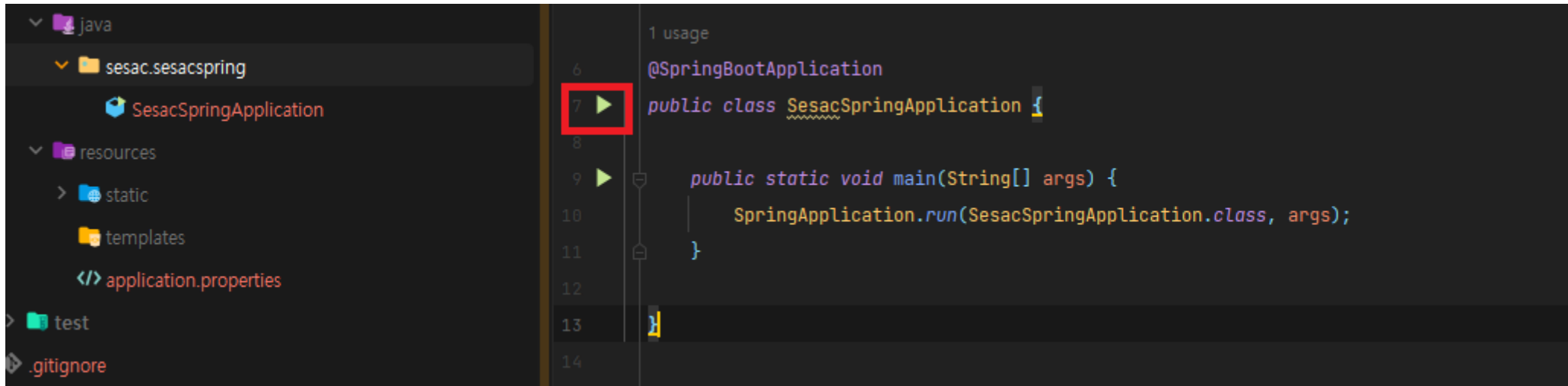
# IntelliJ 프로젝트



# IntelliJ 프로젝트



# IntelliJ 프로젝트 실행



## Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

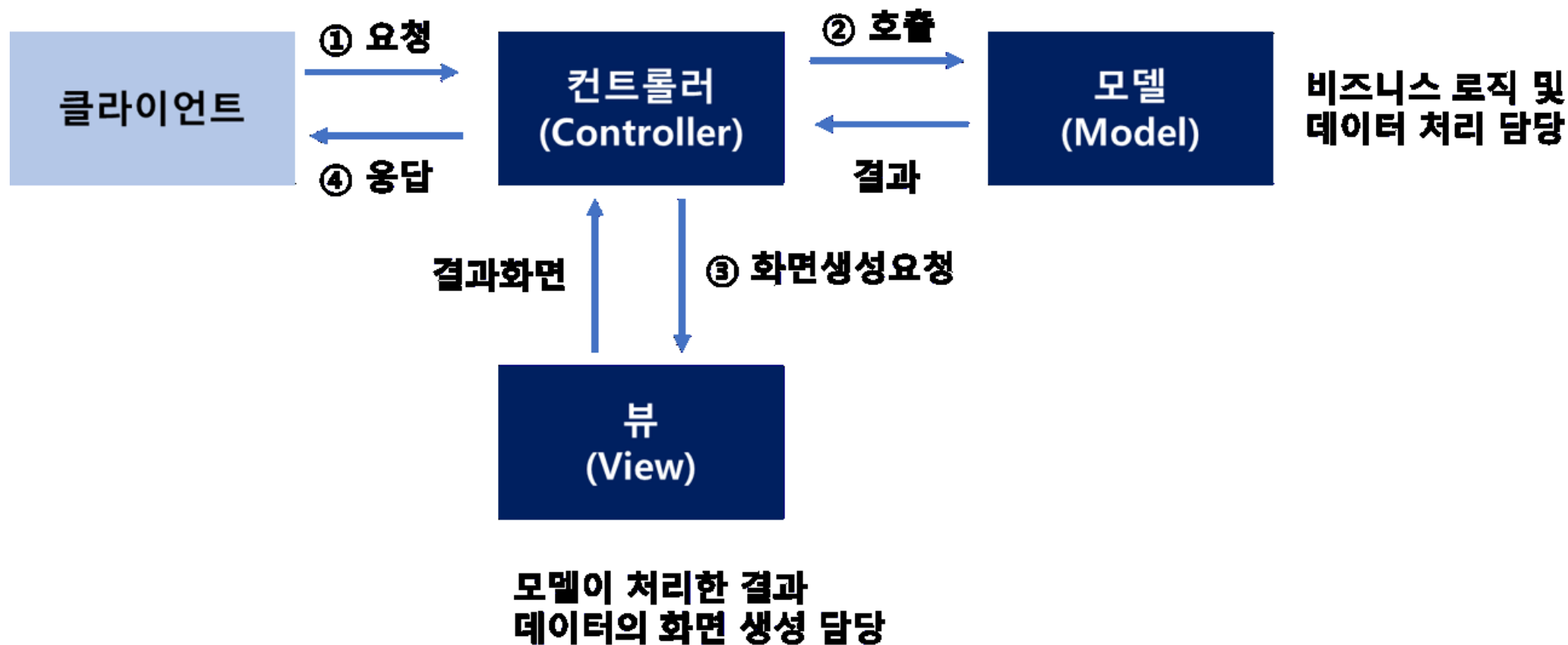
Mon Nov 07 16:19:12 KST 2022

There was an unexpected error (type=Not Found, status=404).

# Spring MVC



# Spring MVC

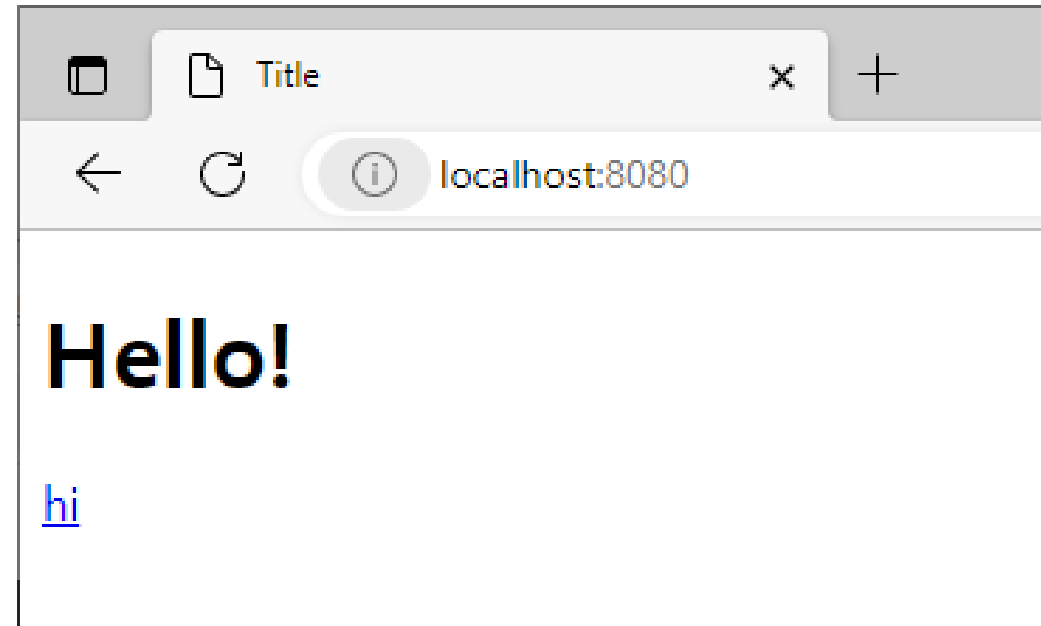


# 정적파일

# 실행 화면 보여주기

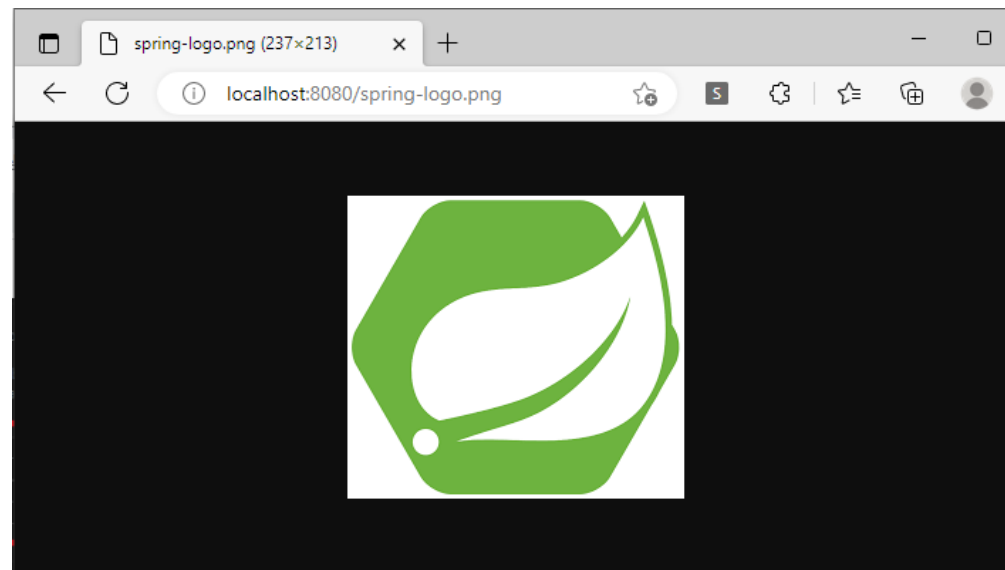
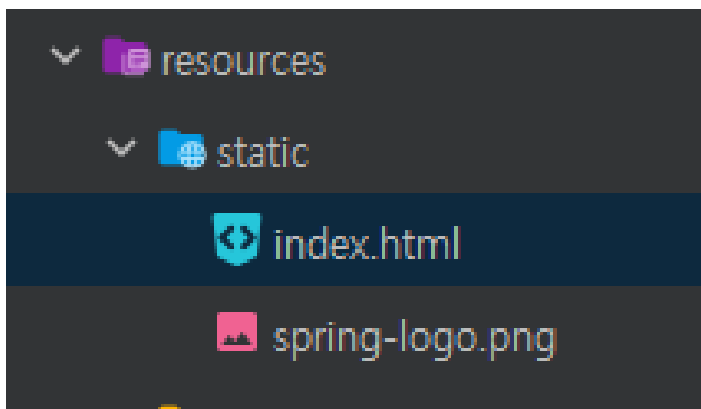
- ./src/main/resources/static에 index.html 생성

```
<!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

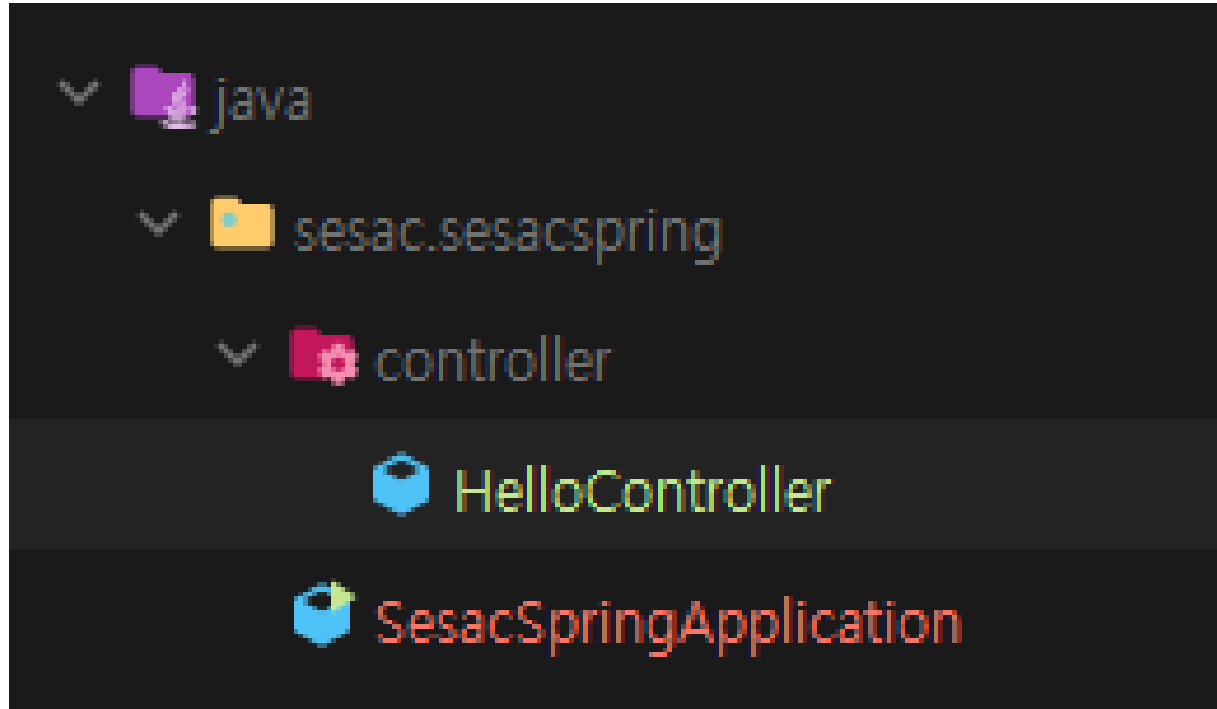
# Thymeleaf란?

- 템플릿 엔진의 일종으로, html 태그에 속성을 추가해 페이지에 동적으로 값을 추가하거나 처리할 수 있게 도와주는 것
- Spring Boot 사용시 권장되는 템플릿 엔진



*Thymeleaf*

# Controller 만들기



# Controller 만들기

```
package sesac.sesacspring.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

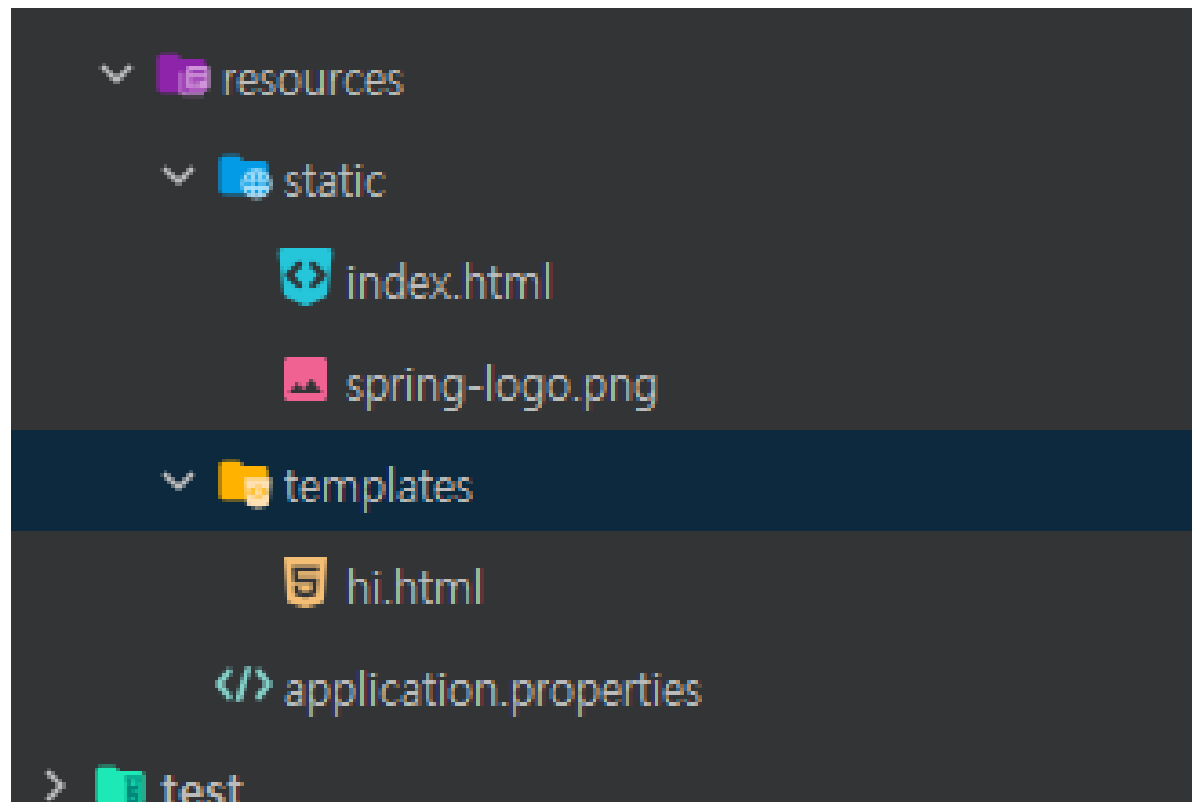
- 해당 클래스가 Controller 클래스라는 것을 Spring Container에게 알려준다.

## @GetMapping

- URL을 매핑시켜주는 것으로 get method로 해당 경로로 들어올 시 getHi 라는 함수를 실행시킨다.



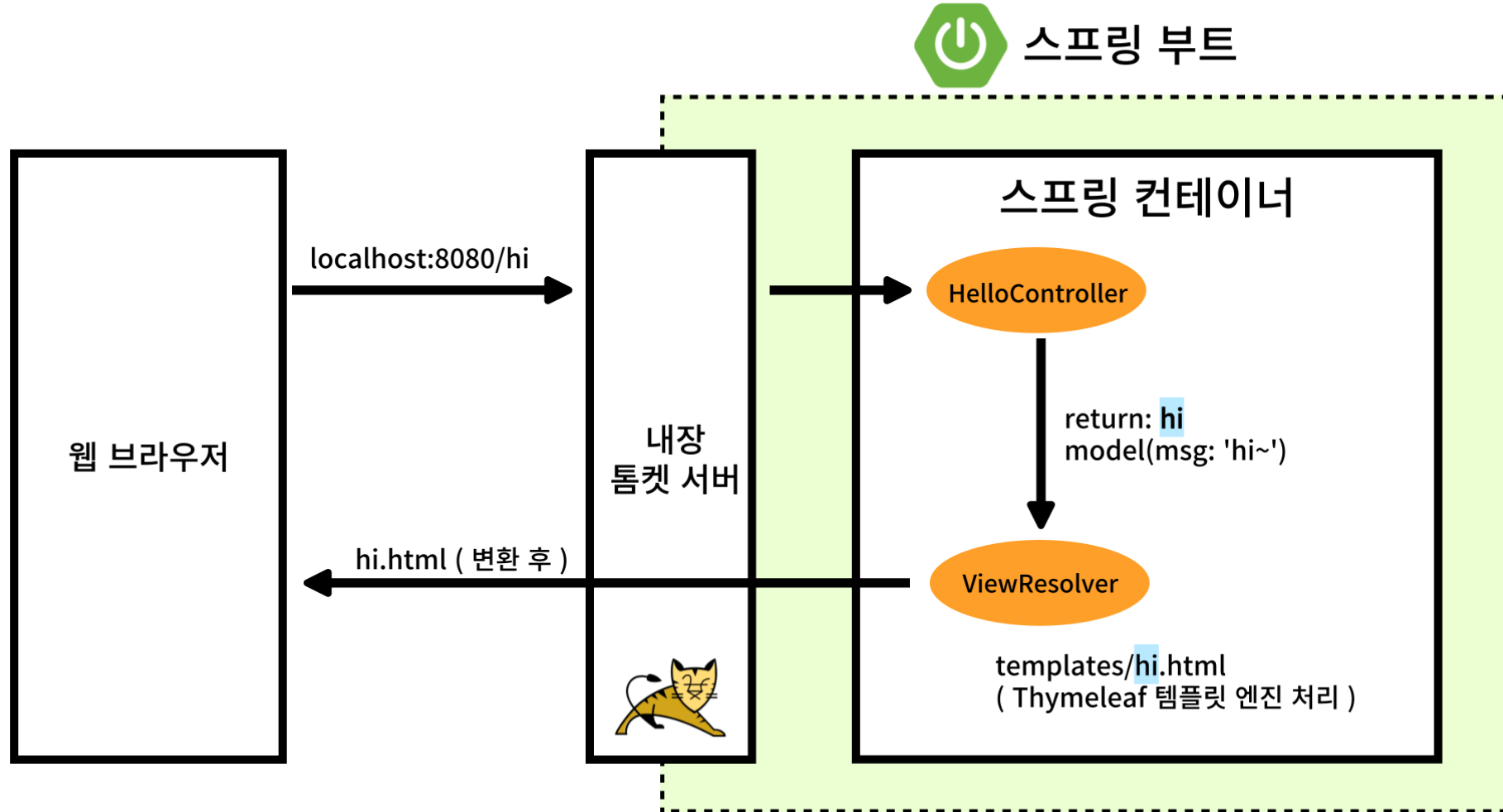
# Template view 만들기



# 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



# Thymeleaf 문법

# Thymeleaf 기본 문법

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

- html 태그 안에 소 문법을 추가한다.
- 이때, 태그는 div 뿐만이 아니라 html에서 지원하는 태그들 모두 사용 가능하다.

# 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 태그값을 반영해서 표시해준다.
- “<strong>안녕</strong>” 값이 서버에서 넘어왔다면 <strong> 이라는 글자가 나오는 것이 아닌 “안녕”이라는 글자가 <strong> 태그에 의해 굵게 표시된다.

# th:value

- Element 들의 value 값을 지정할 때 사용된다.

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



# th:with

- 변수 값을 지정해서 사용하고자 할 때 사용된다.

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

- 위와 같이 사용한다면 서버에서 전달받은 'hello' 값이 저장된 temp 라는 변수가 만들어지고, Thymeleaf 문법 내에서 사용 가능하다.

# th:switch

- Switch-case 문을 이용할 때 사용되는 문법
- th:case 에서 case 문을 다루고, \* 로 case문에서 다루지 않은 모든 경우가 처리된다. ( \* 는 일반 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 사용한다.

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

- th:unless 에 if에 적은 조건을 적어줘야 위의 if가 아닌 경우를 인식할 수 있다는 것 **주의!**

# th:each

- 반복문이 필요한 경우에 사용되는 문법
- 리스트와 같은 collection 자료형을 서버에서 넘겨주면 그에 맞춰 반복적인 작업이 이루어질 때 사용된다.

```
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 1. Thymeleaf (1)

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

10세는 미성년자입니다.

2. Age가 20보다 크다면 아래와 같이 출력

20세는 성인입니다.

## 실습 2. 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