IntelliJ로 시작하는 스프링 입문 공부

유채린

chaerin.du.ub@gmail.com

해당 내용은 인프런 강의를 정리한 내용입니다.
잘못된 내용에 대하여 피드백주시면 감사하겠습니다!

(인프런 아이콘 클릭 시, 참고한 인프런 강의 사이트로 이동합니다.)

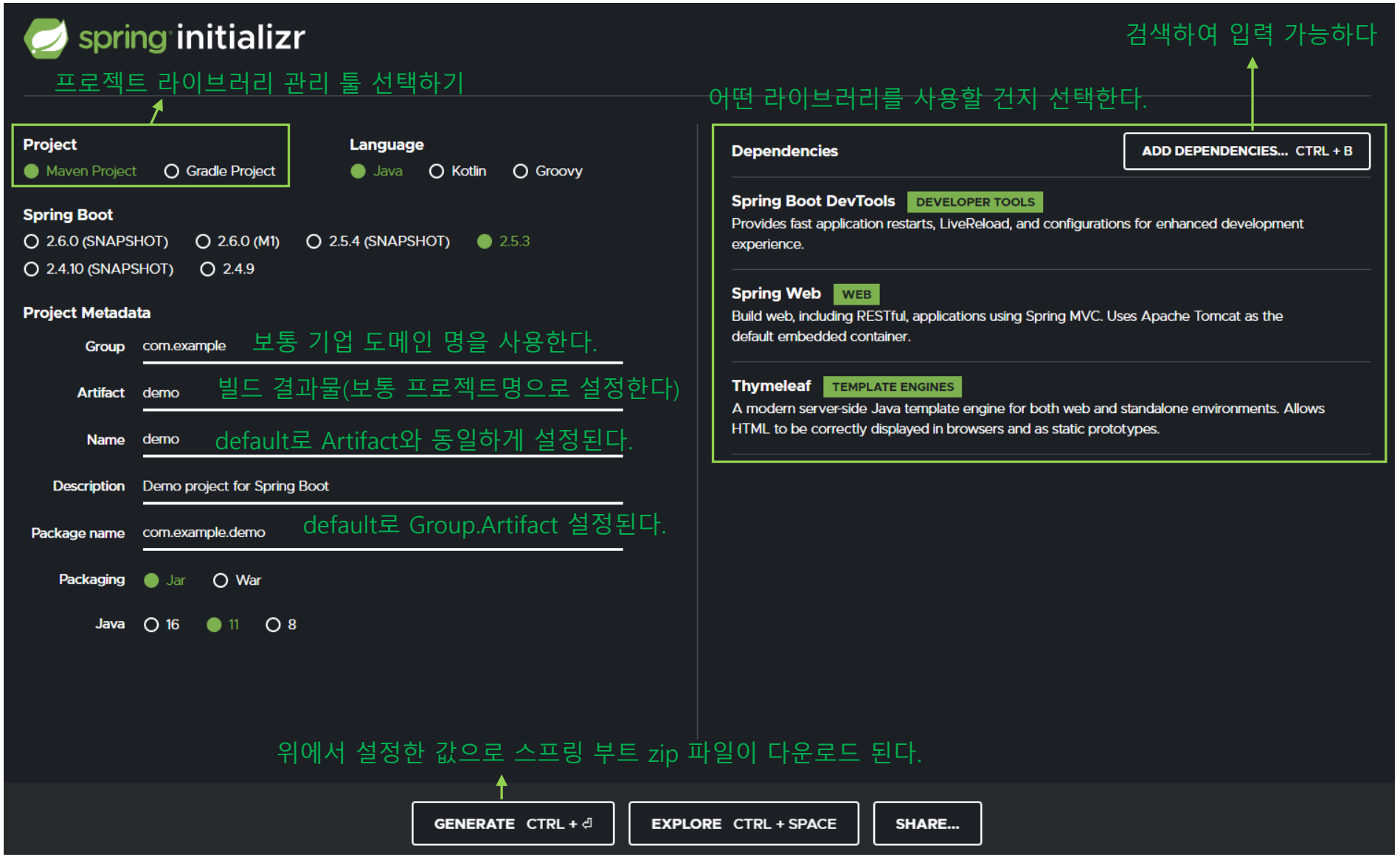
Index

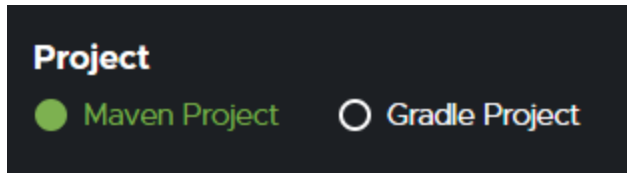
- 프로젝트 환경 설정
- 스프링 웹 개발 기초
- 강의를 들으며 알게 된 단축키

프로젝트 환경 설정

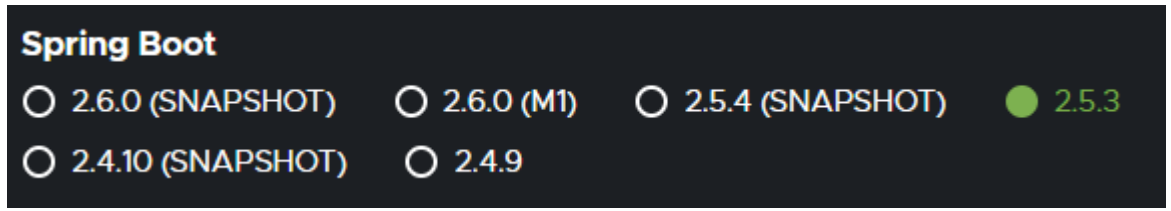
프로젝트 생성, 라이브러리, view 환경 설정, 빌드/실행

start.spring.io 사이트를 이용하여 스프링 **프로젝트 생성**하기

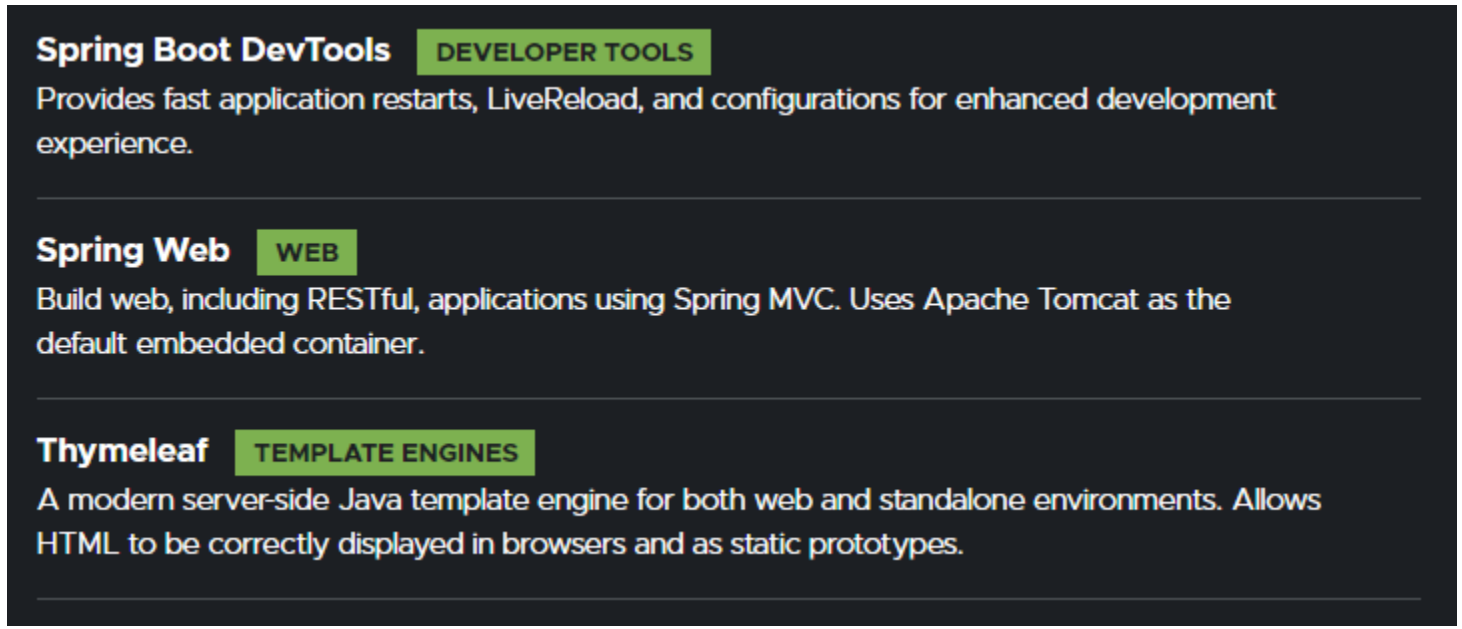




필요한 라이브러리를 갖고 오고 관리해주는 툴
요즘은 gradle을 많이 사용하며,
스프링 라이브러리 관리 자체도 gradle을 이용한다고 한다.



SNAPSHOT: 현재 개발 중인 버전
M1: 아직 정식 릴리즈가 안 된 경우
둘 다 베타버전으로 생각해서 오류 발생 감안해서 사용해야 되는 듯 하다.
안전하게 사이트 접속 시, default로 선택된 항목으로 선택하기

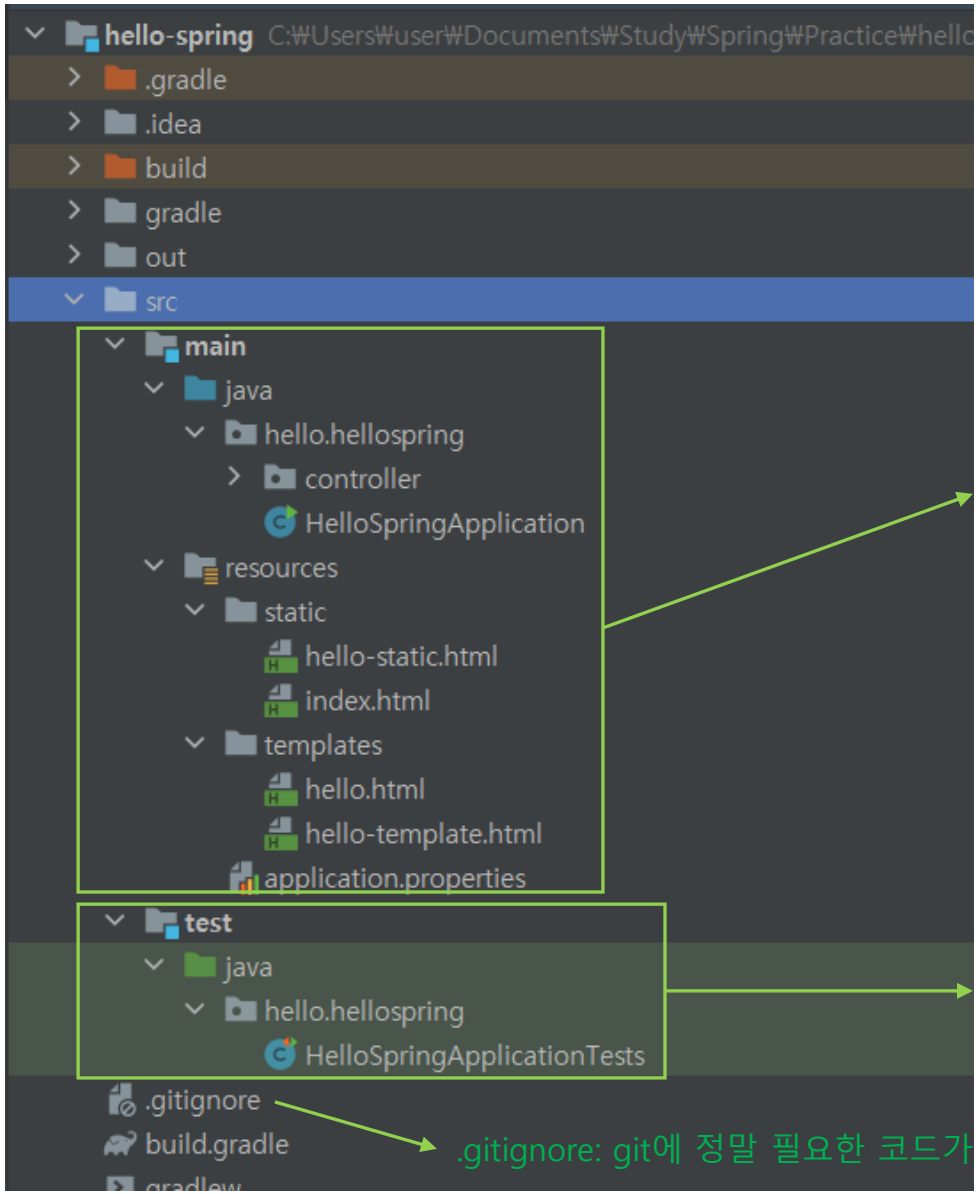


Spring Boot DevTools
: html 파일을 컴파일만 해주면 서버 재시작없이 View파일 변경이 가능하다.
(컴파일: Build > Recompile)

Spring Web
: web project를 만들기 위해서 해당 라이브러리가 필요하다.

Thymeleaf
: html 만들어주는 템플릿 엔진으로 회사마다 사용하는 라이브러리가 다르다.
Ex) freemarker

스프링 부트 zip 파일 압축 해제 후, build.gradle을 선택하여 프로젝트를 open한다.



src

main 폴더
- java(실제 소스들과 패키지)
- resources(실제 자바 코드를 제외한 xml, html, properties 파일들)

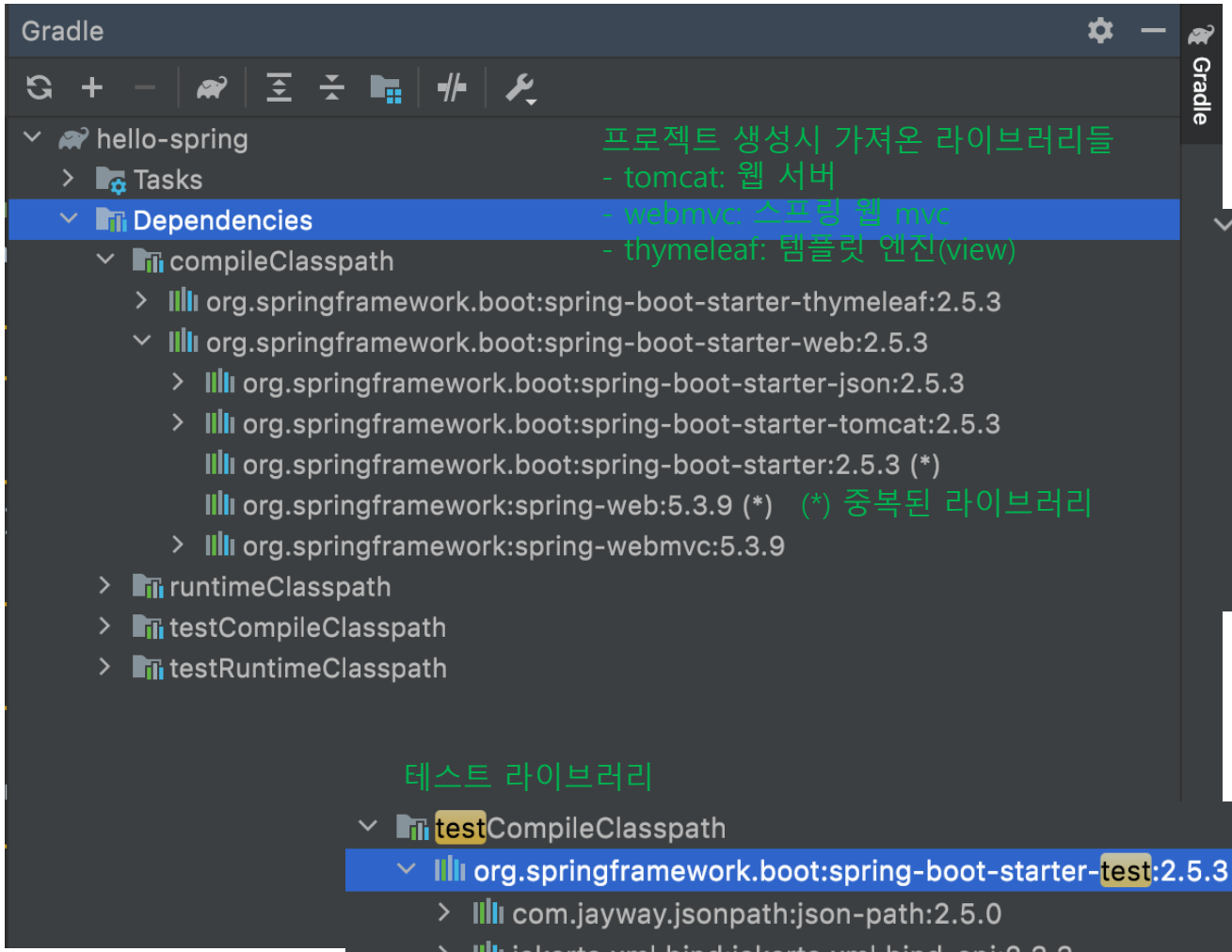
test 폴더
test code들과 관련된 소스들
-> 개발 트렌드

build.gradle

```
plugins {
    id 'org.springframework.boot' version '2.5.3'
    id 'io.spring.dependency-management' version '1.0.11.RELEASE'
    id 'java'
}

group = 'hello'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '11' 자바 소스 컴파일할 수 있는 버전

repositories {
    mavenCentral()   라이브러리 다운받는 곳으로 특정 site를 사용자가 추가(수정)이 가능하다.
}

dependencies {  start.spring.io 사이트에서 선택했던 라이브러리들이 조회된다.
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}           Test library는 기본적으로 들어간다.

test {
    useJUnitPlatform()
}
```

요즘은 소스 라이브러리에서 웹 서버(ex.톰캣)를 갖고 있어서 실행만 해도 웹 서버가 뜬다.

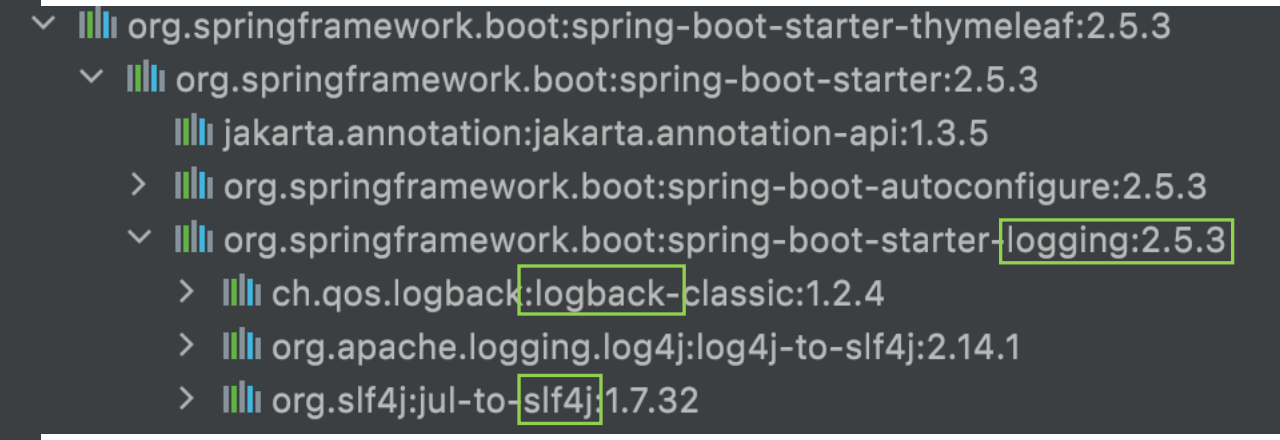
스프링 부트 라이브러리 -> gradle tab에서 조회 가능하다



프로젝트 생성시 가져온 라이브러리들
- tomcat: 웹 서버
- webmvc: 스프링 웹 mvc
- thymeleaf: 템플릿 엔진(view)

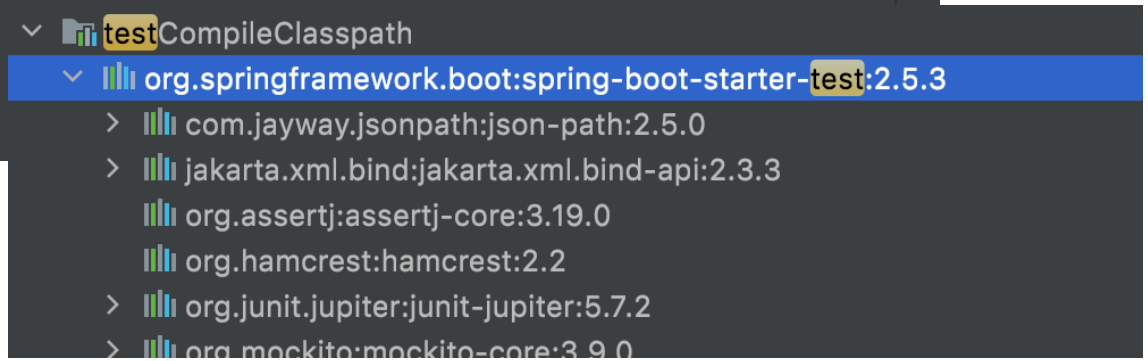
(*) 중복된 라이브러리

기본적으로 들어있는 라이브러리들 (spring-boot-starter)
: 스프링 부트 + 스프링 코어 + 로깅



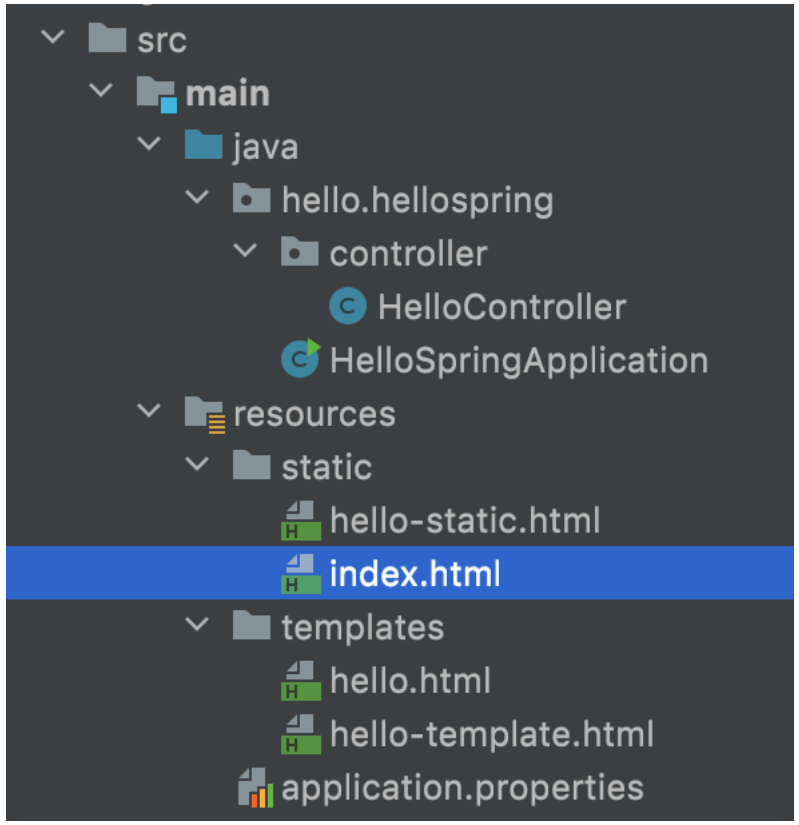
로그는 보통 디버깅할 때나 성능 분석 등 다양한 용도로 사용 가능하다.

테스트 라이브러리



junit: 요즘은 5 버전 많이 사용한다.
mockito
assertj: 테스트 코드 좀 더 편하게 작성 도와주는 라이브러리

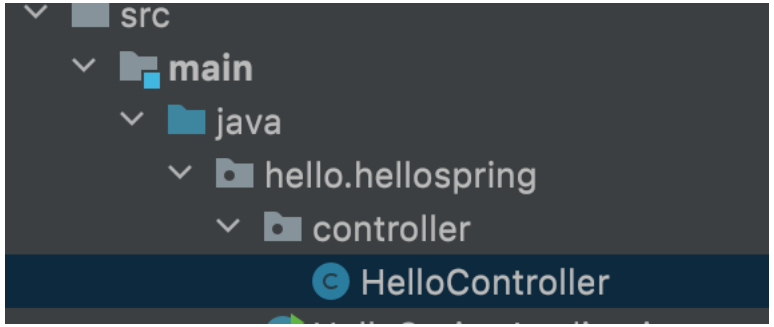
View 환경설정



스프링 부트가 제공하는 welcome page 기능
-> resources > static 디렉토리 위치에 index.html을 생성하면 welcome page가 된다.
(만약 index.html 파일이 없는 경우, index template을 찾아본다.)

스프링 MVC는 다양한 템플릿 기술(freemarker, thymeleaf, jsp)을 지원해주며 다른 템플릿 엔진에는 자체 스프링 MVC 통합 기능이 있다.

스프링 부트는 FreeMarker, Groovy, Thymeleaf 같은 템플릿 엔진에 대한 auto-configuration을 지원한다.



controller 패키지 생성 후,
해당 패키지 아래에 Controller 자바 파일을 생성한다.

HelloController.java

```java
@Controller
public class HelloController {

    @GetMapping("hello")
    public String hello(Model model) {
        model.addAttribute( attributeName: "data", attributeValue: "spring!!!");
        return "hello";
    }
```

컨트롤러 클래스에 무조건 Annotation 추가 필요하다.

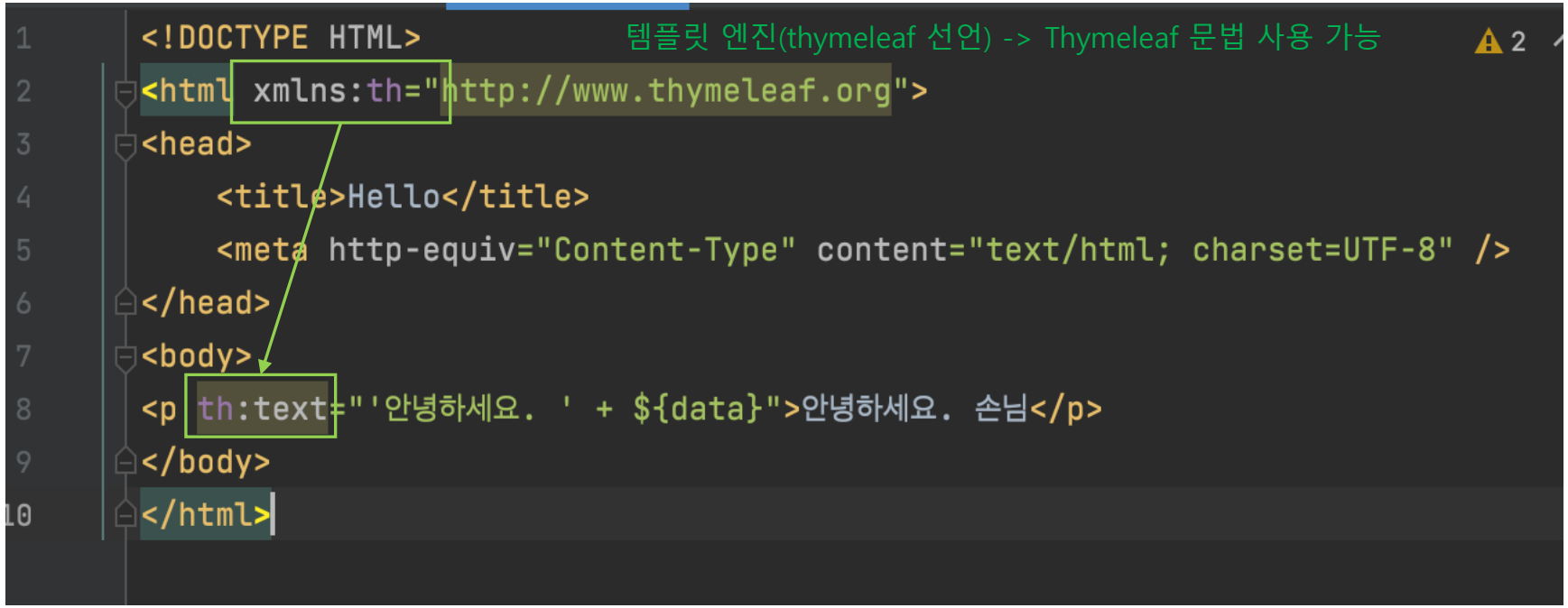
@GetMapping: http protocol 메서드 (get/post)로 이해하면 될 듯 하다.

model.addAttribute(key, value)

스프링 부트가 resources>templates>hello.html 을 자동으로 찾아서 Return

컨트롤러에서 리턴 값으로 문자를 반환하게 되면 **뷰 리졸버(viewResolver)가 화면을 찾아서 처리한다.**
(**resources/templates/ViewName.html**)

resources>template>hello.html



빌드/실행

프로젝트 디렉토리에 들어가서 빌드를 한다. -> **./gradlew build**

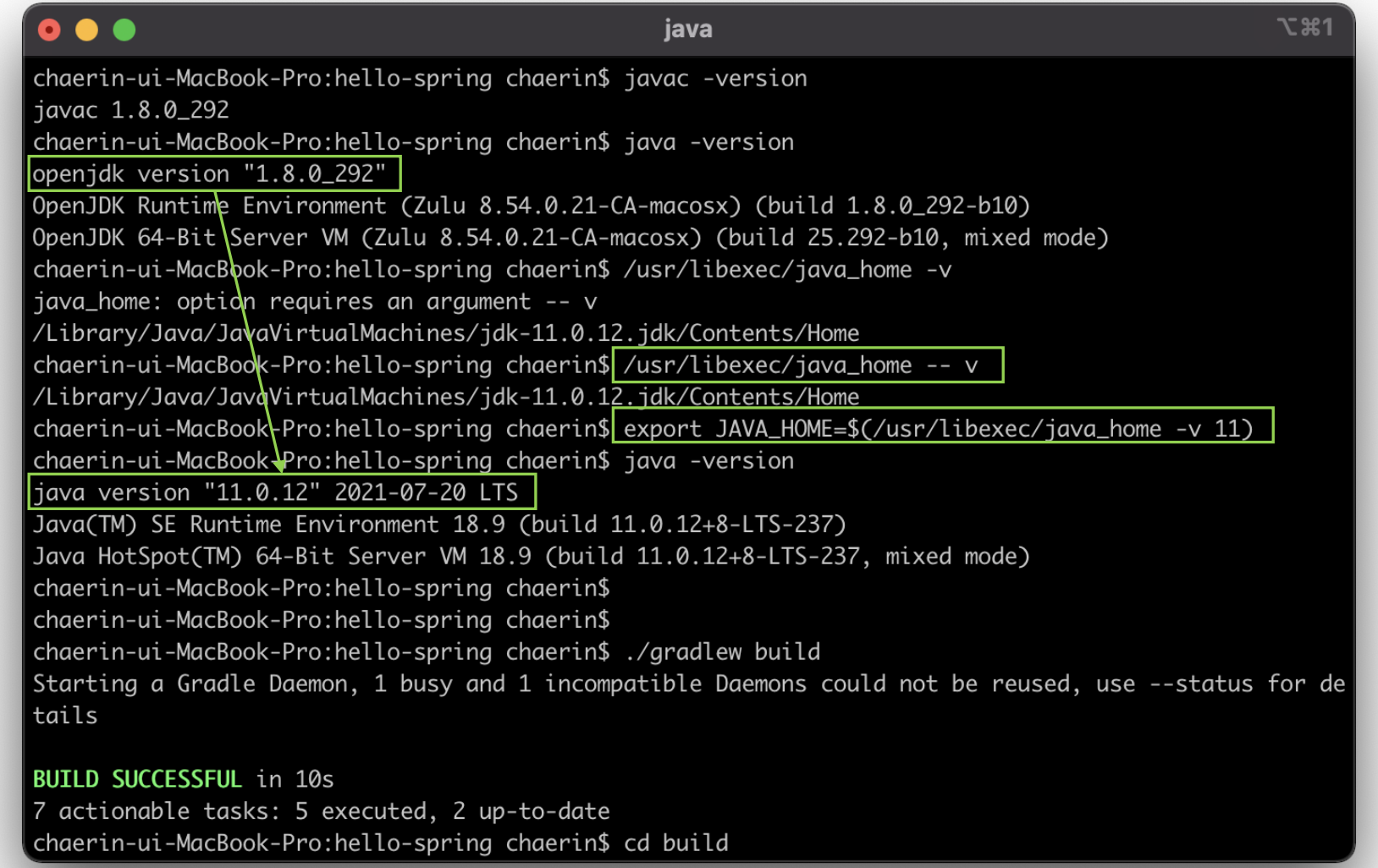
빌드 후, build 디렉토리가 생성이 된다.
build 디렉토리 안의 libs 디렉토리로 들어가게 되면
hello-spring-0.0.1-SNAPSHOT.jar 파일이 있는 걸 조회할 수 있다.
-> **java –jar hello-spring-0.0.1-SNAPSHOT.jar**

빌드 실행이 되면 끝~ (IntelliJ IDE에서 종료 후 해야 한다! 그렇지 않으면 이미 사용 중이라 안 됨)

** 빌드 잘 안 될 경우,
./gradlew clean 후, (build 디렉토리 삭제됨)
./gradlew clean build

Task :compileJava FAILED 뜨는 경우

jdk버전이 프로젝트 jdk버전과 같지 않은 경우로 재설정 해준다.



스프링 웹 개발 기초

정적 컨텐츠, MVC와 템플릿 엔진, API

- 정적 컨텐츠

: welcome page처럼 서버에서 하는 것 없이 파일을 웹 브라우저에 그대로 내려주는 것
스트링 부트는 정적 컨텐츠 기능을 자동으로 제공한다.

- MVC와 템플릿 엔진

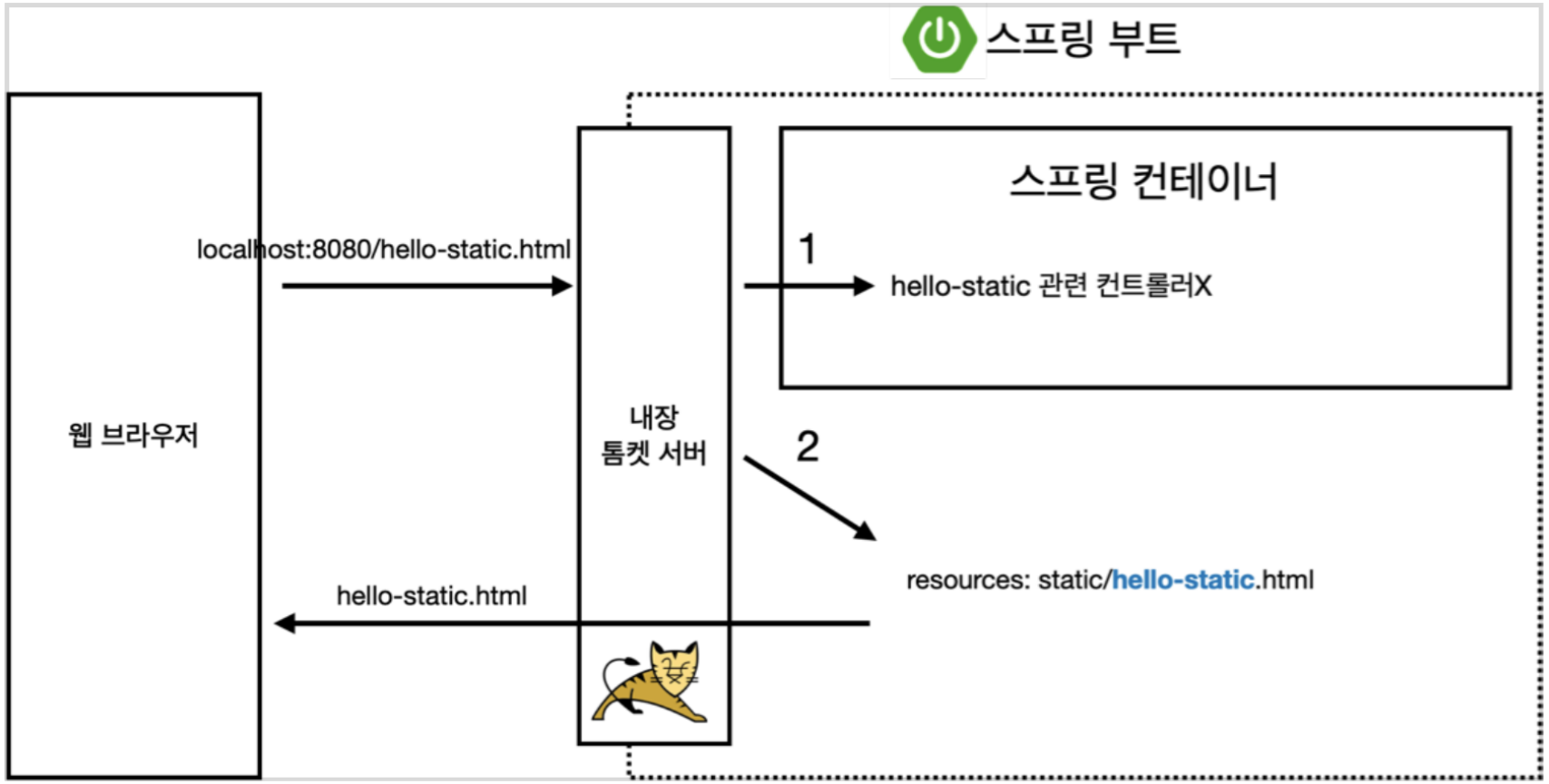
: 서버에서 프로그램을 하여 HTML을 동적으로 주는 것

- API

: 보통 json 데이터 구조로 데이터를 전달하는 방식

정적 컨텐츠

resources>static>hello-static.html 이라는 파일 생성 시,
서버가 빌드 되면 해당 디렉토리의 정적 컨텐츠 호출이 가능하다.
(html파일만으로도 조회 가능)



톰캣 서버가 요청을 받은 후, 스프링에게 넘겨준다.
스프링은 Controller 우선순위로 확인을 하고,
없는 경우 스프링 부트가 resources안에 있는 static/hello-static.html을 찾아와 반환해준다.

MVC와 템플릿 엔진

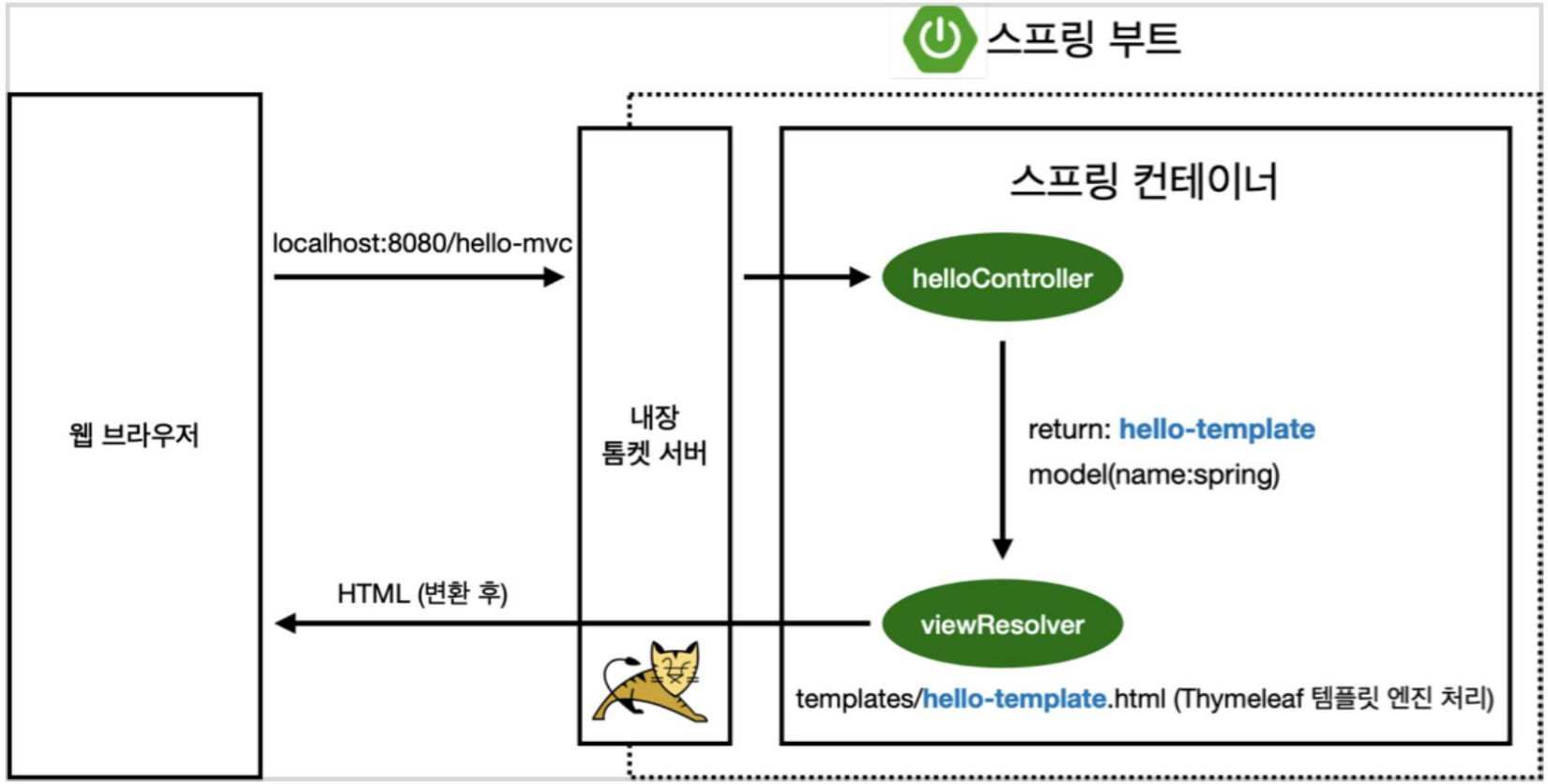
```java
@GetMapping("hello-mvc")
public String helloMvc(@RequestParam(value="name",required = false) String name, Model model) {
    model.addAttribute( attributeName: "name", name);
    return "hello-template";
}
```

required default 값은 true로, 값을 무조건 넘겨줘야 에러가 안 난다.
False로 설정하여 값 넘겨주지 않을 경우 null로 뜬다.

hello-template.html × resources > templates > hello-template.html

```html
1  <html xmlns:th="http://www.thymeleaf.org">
2  <body>
3  <p th:text="'hello ' + ${name}">hello! empty</p>
4  </body>
5  </html>
```

MVC, 템플릿 엔진 이미지

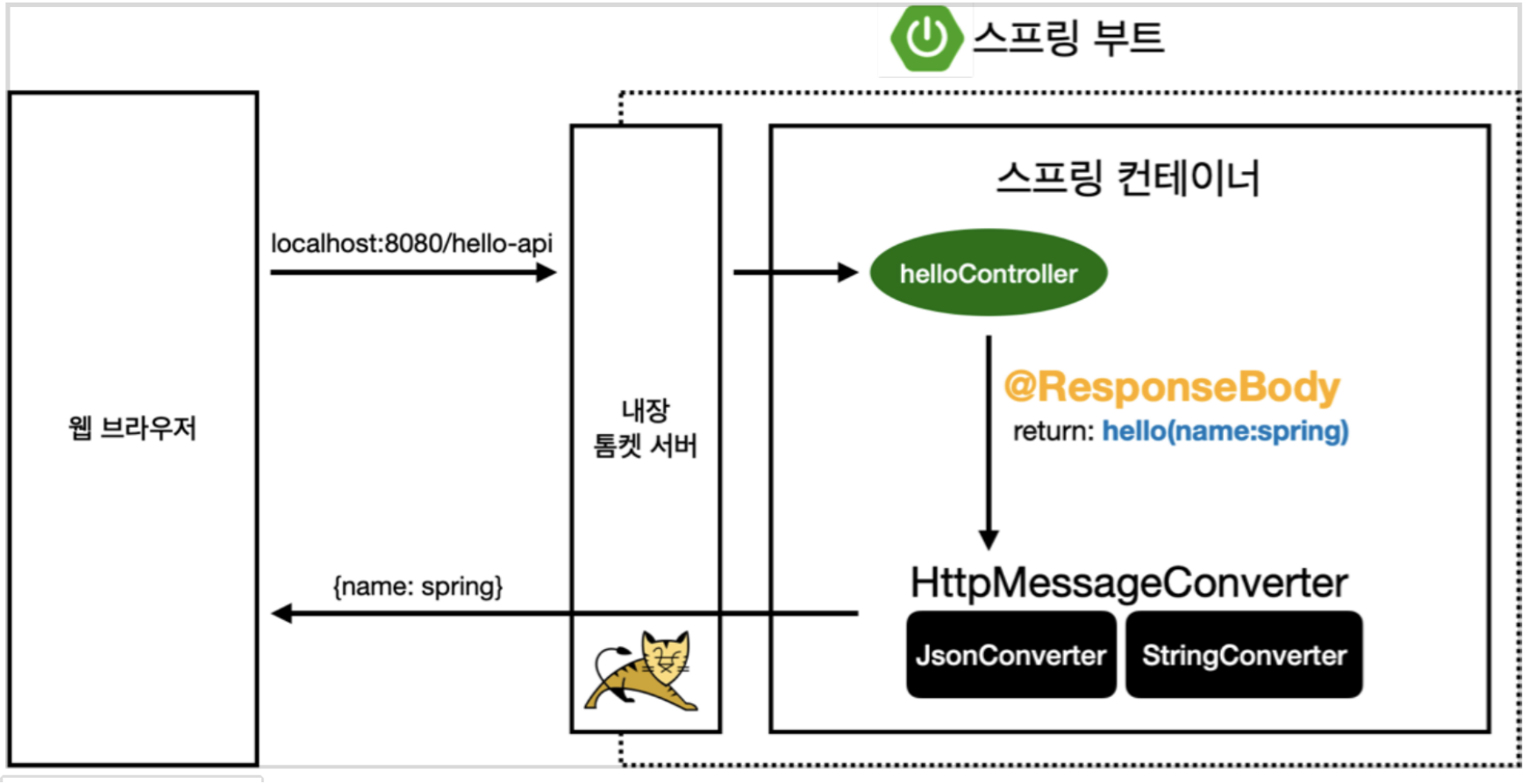


내장 톰캣서버가 요청을 먼저 받아 스프링에게 넘겨준다.
스프링 부트는 컨트롤러를 확인하고 “hello-template”으로 반환해준다.
(viewResolver: 뷰를 찾아주고 템플릿 엔진 연결해줌)
viewResolver가 hello-template.html을 찾아서 thymeleaf 템플릿 엔진에게 넘겨주고
템플릿 엔진은 렌더링하여 html 변환 후 넘겨준다. (정적 컨텐츠는 그대로 넘겨줬었다.)

API

```java
@GetMapping("hello-string")
@ResponseBody
public String helloString(@RequestParam("name") String name) { return "hello " + name; }

@GetMapping("hello-api")
@ResponseBody
public Hello helloApi(@RequestParam("name") String name) {
    Hello hello = new Hello();
    hello.setName(name);
    return hello;
}
static class Hello {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

html의 body 태그가 아닌 http 응답
-> viewResolver를 사용하지 않기 때문에 직접 반환해준다.



내장 톰캣 서버가 요청을 먼저 받아 스프링에게 넘겨준다.

스프링 부트는 컨트롤러를 확인한다. -> @ResponseBody가 붙어 있는 경우, http 응답에 그대로 데이터를 넘겨줘야 한다고 판단한다. 객체를 반환하면 객체가 json으로 변환된다.

단순 문자열일 경우 StringConverter, 객체라면 jsonConverter가 실행된다.

(MVC에서는 viewResolver)

단축키

	Mac	**Window**
Project Structure	Command + ;	Ctrl + Alt + Shift + S
Settings	Command + ,	Ctrl + Alt + S
Java method parameter 정보 조회	Command + P	Ctrl + P
자동완성	Command + Shift + Enter	Ctrl + Shift + Enter
Getter/setter/constructor ..	Command + N	Ctrl + N

감사합니다