# 1. 프로젝트 환경 설정

#1.인강/1.스프링 입문/강의#

- /프로젝트 생성
- /라이브러리 살펴보기
- /View 환경설정
- /빌드하고 실행하기

## 프로젝트 생성

**사전 준비물**

- **Java 17 이상 설치**
- IDE: IntelliJ 또는 Eclipse 설치

> **주의! 스프링 부트 3.0 이상, JDK 17 이상을 사용해야 합니다.**

**스프링 부트 스타터 사이트로 이동해서 스프링 프로젝트 생성**

https://start.spring.io

- 프로젝트 선택
  - Project: **Gradle - Groovy** Project
  - Spring Boot: **3.x.x**
  - Language: Java
  - Packaging: Jar
  - Java: 17 또는 21
- Project Metadata
  - groupId: hello
  - artifactId: hello-spring
- Dependencies: Spring Web, Thymeleaf

### 주의! - 스프링 부트 3.x 버전 선택 필수

**start.spring.io 사이트에서 스프링 부트 2.x에 대한 지원이 종료되어서 더는 선택할 수 없습니다.**

**이제는 스프링 부트 3.0 이상을 선택해주세요.**

스프링 부트 3.0을 선택하게 되면 다음 부분을 꼭 확인해주세요.

- **1. Java 17 이상**을 사용해야 합니다.
- **2. javax 패키지 이름을 jakarta로 변경**해야 합니다.
  - 오라클과 자바 라이센스 문제로 모든 `javax` 패키지를 `jakarta`로 변경하기로 했습니다.
- **3. H2 데이터베이스를 2.1.214 버전 이상 사용해주세요.**

**패키지 이름 변경 예)**

- **JPA 애노테이션**
  - `javax.persistence.Entity` ➜ `jakarta.persistence.Entity`
- **스프링에서 자주 사용하는 @PostConstruct 애노테이션**
  - `javax.annotation.PostConstruct` ➜ `jakarta.annotation.PostConstruct`
- **스프링에서 자주 사용하는 검증 애노테이션**
  - `javax.validation` ➜ `jakarta.validation`

스프링 부트 3.x 관련 자세한 내용은 다음 링크를 확인해주세요: https://bit.ly/springboot3

> **참고**
> 지금은 영상을 찍던 시점의 2.3.1 버전이 선택지에 없습니다.
> Spring Boot 버전은 SNAPSHOT, M1 같은 미정식 버전을 제외하고 최신 버전을 사용하시면 됩니다.
> 예) 2.7.1 (SNAPSHOT) ➜ 이것은 아직 정식 버전이 아니므로 선택하면 안됩니다.
> 예) 2.7.0 ➜ 이렇게 뒤에 영어가 붙어있지 않으면 정식 버전이므로 이 중에 최신 버전을 선택하면 됩니다.

**Gradle 전체 설정**

`build.gradle`

```
plugins {
    id 'org.springframework.boot' version '2.3.1.RELEASE'
    id 'io.spring.dependency-management' version '1.0.9.RELEASE'
    id 'java'
}

group = 'hello'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '11'

repositories {
    mavenCentral()
}
```

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    testImplementation('org.springframework.boot:spring-boot-starter-test') {
        exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'
    }
}

test {
    useJUnitPlatform()
}
```

- 동작 확인
  - 기본 메인 클래스 실행
  - 스프링 부트 메인 실행 후 에러페이지로 간단하게 동작 확인(`http://localhost:8080`)

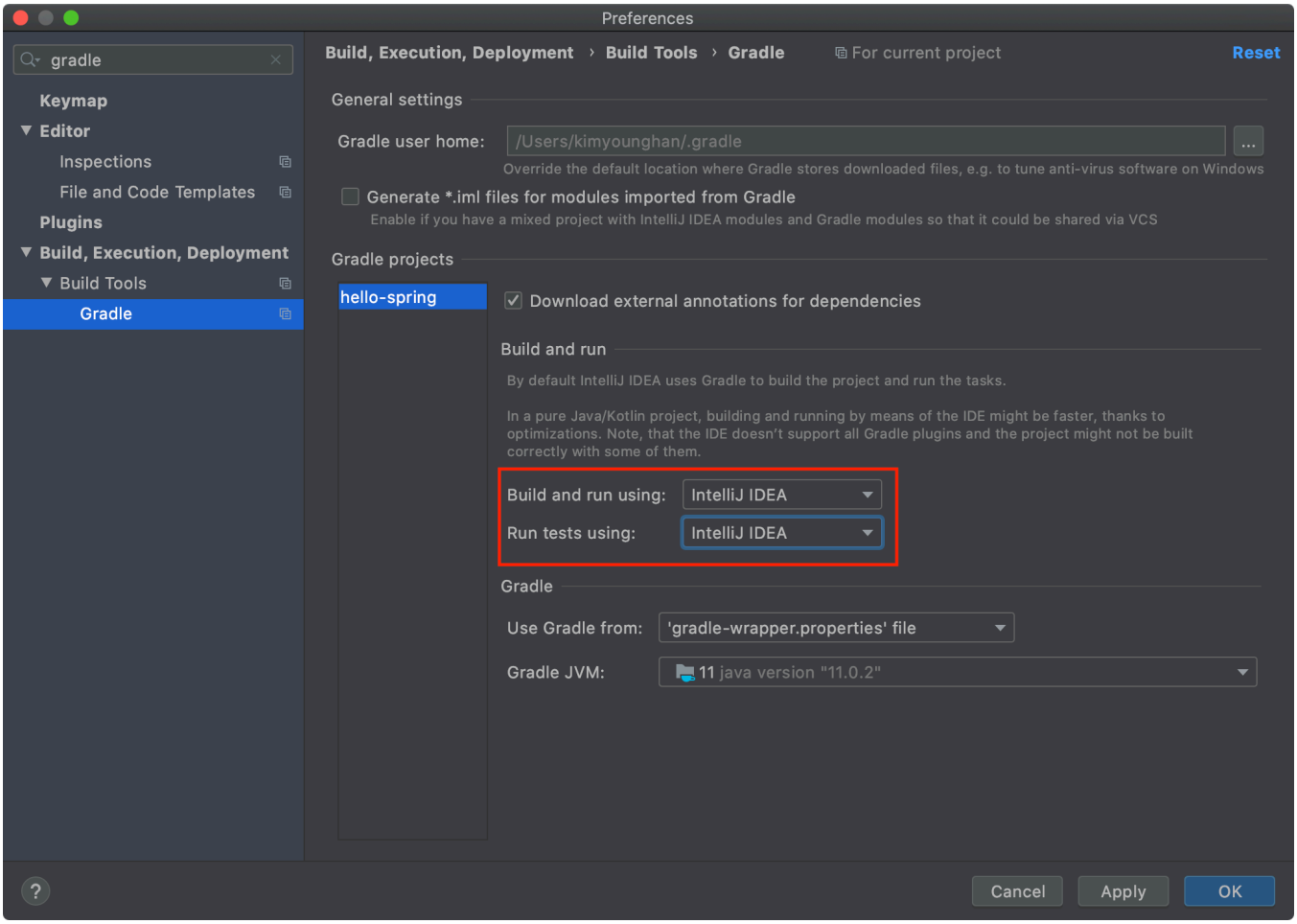
## IntelliJ Gradle 대신에 자바 직접 실행

최근 IntelliJ 버전은 Gradle을 통해서 실행 하는 것이 기본 설정이다. 이렇게 하면 실행속도가 느리다. 다음과 같이 변경하면 자바로 바로 실행해서 실행속도가 더 빠르다.

- Preferences ➜ Build, Execution, Deployment ➜ Build Tools ➜ Gradle
  - Build and run using: Gradle ➜ IntelliJ IDEA
  - Run tests using: Gradle ➜ IntelliJ IDEA

> **윈도우 사용자** File ➜ Setting

**설정 이미지**




## 윈도우 사용자를 위한 IntelliJ 단축키 조회 방법

### 윈도우 단축키 확인 법

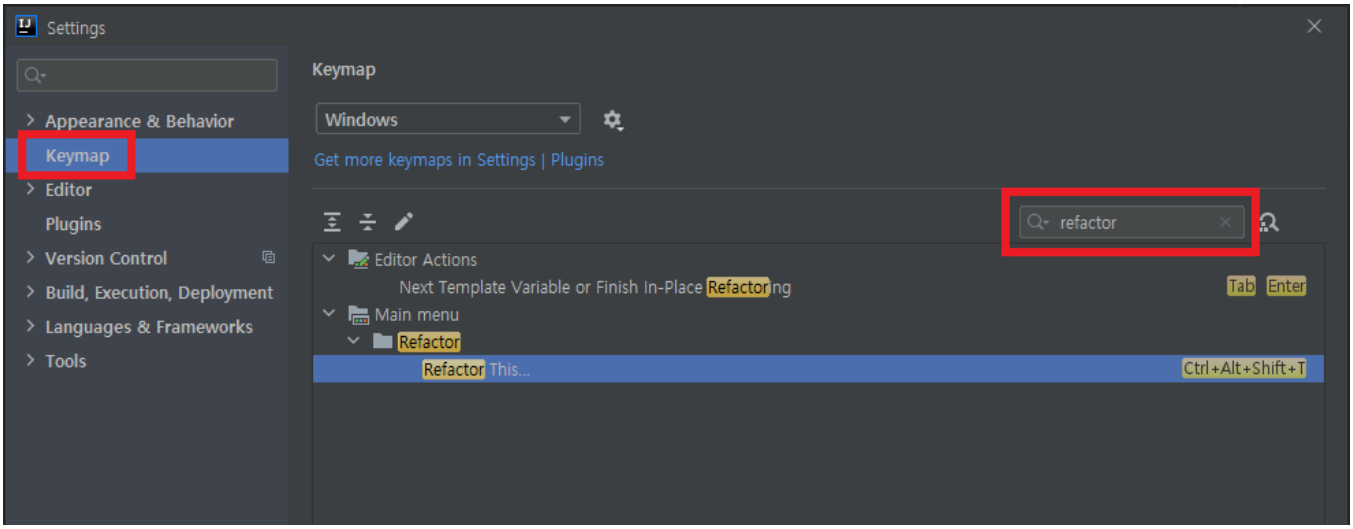
단축키는 영상 화면 아래쪽에 보면 오른쪽 괄호안에 윈도우용 단축키도 나옵니다.


표기가 좀 어려울 수 있는데요. 이 단축키는 윈도우에서 다음 키에 대응합니다.

**Ctrl + Alt + Shift + T**

### IntelliJ에서 단축키를 확실하게 검색하는 방법

- File ➜ Settings에 들어간다.
- 다음 화면 왼쪽에 보이는 것 처럼 keymap을 선택한다.
- 다음 화면 오른쪽에 있는 검색창에 단축키 이름을 입력한다. 단축키 이름은 위 그림 처럼 영상 하단에 나온다.
- 다음 그림을 보면 Refactor This의 윈도우 단축키는 Ctrl + Alt + Shift + T 인 것을 알 수 있다.




## IntelliJ JDK 설치 확인

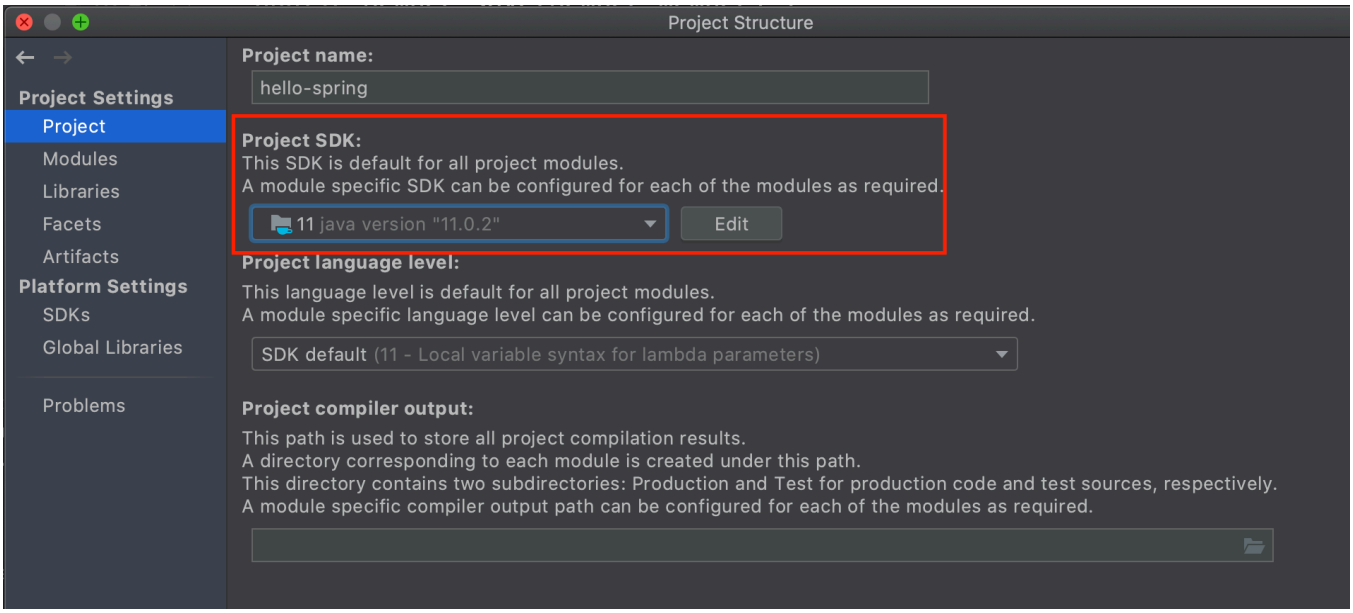
> **주의!** JDK 17 버전 이상을 설치해주세요. 다른 버전을 설치하면 정상 동작하지 않을 가능성이 높습니다.

IntelliJ에서 자바 실행이 잘 안되면 다음 부분을 확인해주세요.(일반적으로 자동으로 설정이 되어 있지만, 가끔 문제가 되는 경우에 참고하시면 됩니다.)

- 프로젝트 JDK 설정
- gradle JDK 설정

먼저 IntelliJ에서 프로젝트 JDK 설정을 확인해주세요.

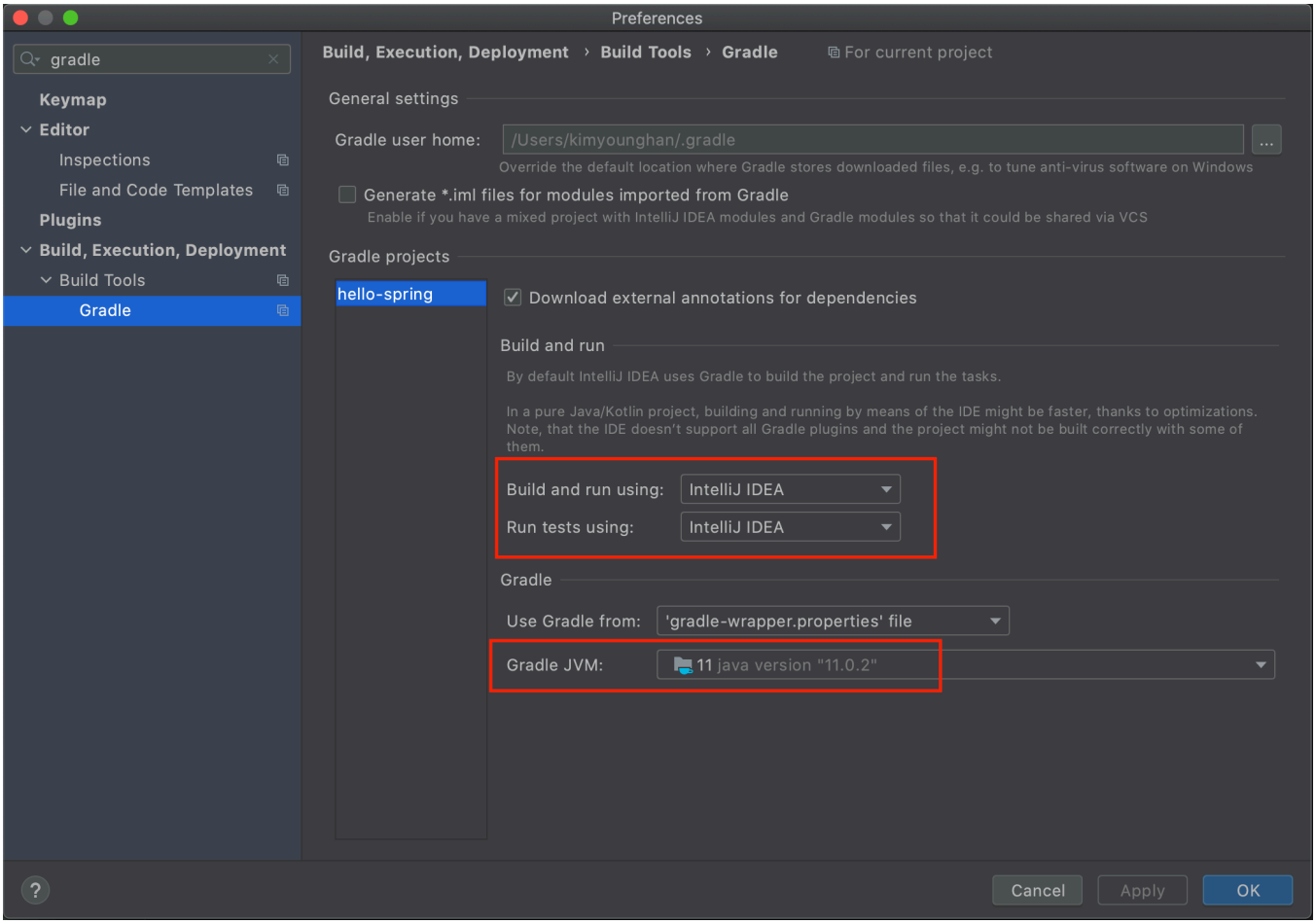
### 프로젝트 JDK 설정



- 다음으로 이동합니다.
  - Windows: File ➜ Project Structure(Ctrl+Alt+Shift+S)
  - Mac: File ➜ Project Structure (⌘;)
- 빨간색 박스의 JDK를 내가 새로 설치한 자바 17 버전 이상으로 지정해줍니다.

다음으로 Gradle이 사용하는 JDK 설정도 확인해주세요.

**Gradle JDK 설정**




- 다음으로 이동합니다.
  - Windows: File ➜ Settings(Ctrl+Alt+S)
  - Mac: IntelliJ IDEA | Preferences(⌘,)
- 발간색 박스의 Build and run using를 IntelliJ IDEA로 선택합니다.
- 발간색 박스의 Build tests using를 IntelliJ IDEA로 선택합니다.
- 발간색 박스 Gradle JVM을 새로 설치한 자바 17 버전 이상으로 지정해줍니다.

# 라이브러리 살펴보기

> Gradle은 의존관계가 있는 라이브러리를 함께 다운로드 한다.

**스프링 부트 라이브러리**

- spring-boot-starter-web
  - spring-boot-starter-tomcat: 톰캣 (웹서버)
  - spring-webmvc: 스프링 웹 MVC
- spring-boot-starter-thymeleaf: 타임리프 템플릿 엔진(View)
- spring-boot-starter(공통): 스프링 부트 + 스프링 코어 + 로깅
  - spring-boot
    - spring-core
  - spring-boot-starter-logging
    - logback, slf4j

**테스트 라이브러리**

- spring-boot-starter-test
  - junit: 테스트 프레임워크
  - mockito: 목 라이브러리
  - assertj: 테스트 코드를 좀 더 편하게 작성하게 도와주는 라이브러리
  - spring-test: 스프링 통합 테스트 지원

# View 환경설정

## Welcome Page 만들기

`resources/static/index.html`

```
<!DOCTYPE HTML>
<html>
<head>
    <title>Hello</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
Hello
<a href="/hello">hello</a>
```

```
</body>
</html>
```

- 스프링 부트가 제공하는 Welcome Page 기능
  - `static/index.html`을 올려두면 Welcome page 기능을 제공한다.
  - https://docs.spring.io/spring-boot/docs/2.3.1.RELEASE/reference/html/spring-boot-features.html#boot-features-spring-mvc-welcome-page

## thymeleaf 템플릿 엔진

- thymeleaf 공식 사이트: https://www.thymeleaf.org/
- 스프링 공식 튜토리얼: https://spring.io/guides/gs/serving-web-content/
- 스프링부트 메뉴얼: https://docs.spring.io/spring-boot/docs/2.3.1.RELEASE/reference/html/spring-boot-features.html#boot-features-spring-mvc-template-engines

```
@Controller
public class HelloController {

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

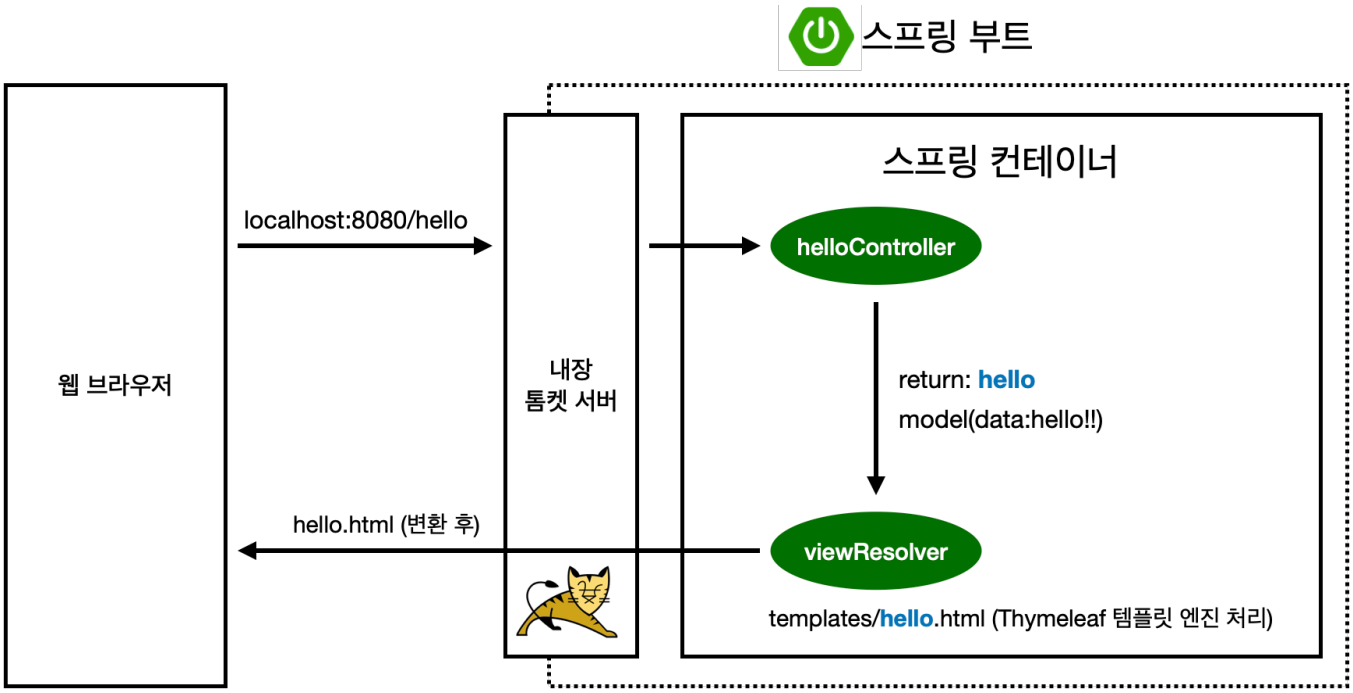
`resources/templates/hello.html`

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Hello</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
<p th:text="'안녕하세요. ' + ${data}" >안녕하세요. 손님</p>
</body>
</html>
```

**thymeleaf 템플릿엔진 동작 확인**

- 실행: http://localhost:8080/hello

**동작 환경 그림**




- 컨트롤러에서 리턴 값으로 문자를 반환하면 뷰 리졸버( `viewResolver` )가 화면을 찾아서 처리한다.
  - 스프링 부트 템플릿엔진 기본 viewName 매핑
  - `resources:templates/` +{ViewName}+ `.html`

> 참고: `spring-boot-devtools` 라이브러리를 추가하면, `html` 파일을 컴파일만 해주면 서버 재시작 없이 View 파일 변경이 가능하다.
> 인텔리J 컴파일 방법: 메뉴 build → Recompile

# 빌드하고 실행하기

콘솔로 이동

1. `./gradlew build`
2. `cd build/libs`
3. `java -jar hello-spring-0.0.1-SNAPSHOT.jar`
4. 실행 확인

**윈도우 사용자를 위한 팁**

- 콘솔로 이동 ➜ 명령 프롬프트(cmd)로 이동
- `./gradlew` ➜ `gradlew.bat`를 실행하면 됩니다.
- 명령 프롬프트에서 `gradlew.bat`를 실행하려면 `gradlew`하고 엔터를 치면 됩니다.
- `gradlew build`
- 폴더 목록 확인 `ls` ➜ `dir`
- 윈도우에서 Git bash 터미널 사용하기
  - 링크: https://www.inflearn.com/questions/53961